THE PAPER

FOR OWNERS OF THE COMMODORE PET™ PERSONAL COMPUTER

VOLUME III, ISSUE 2 & 3 AN ARESCO PUBLICATION MARCH/APRIL 1980, $2.50

TABLE OF CONTENTS

ADVERTISERS

FOR YOUR GENERAL INFORMATION

THE PAPER is published ten times per year by ARESCO, Inc., at 6303 Golden Hook, Columbia, MD, 21044. Telephone (301) 730-5186.

THE PAPER is mailed to subscribers during the last week of each month except June and December. Single copy price is $2 and subscription price is $15 for all ten issues of the current volume. Subscriptions do not carry over from one volume to the next. Subscribers who do not wish to order the full volume should send $2 for each issue desired. Renewals are accepted during the last two months of the current volume year, and the first issue of each volume is published in Feb. Subscription orders should be mailed to P O Box 1142, Columbia MD, 21044 in order to assure prompt processing.

Second class postage paid in Columbia MD 21045 (USPS 450-930). POSTMASTER Send all address changes to THE PAPER, Box 1142, Columbia MD 21044

THE PAPER is not associated in any way with Commodore Business Machines, Commodore International, or any of its affiliated companies, and Commodore is not responsible for the contents of THE PAPER.

Readers are encouraged to submit articles of general interest to PET owners for publication in the PAPER. Material submitted must be free of copyright restrictions.

SUBSCRIPTION RATES

USA residents: $15/10 issues of current volume. Non-USA residents should include an additional $12 for air mail postage if desired. Complete sets of previous volumes will be available for $15/set (all ten issues of the volume), plus postage. No purchase orders will be accepted for orders; payment must accompany the application for subscription. COD (via UPS) is acceptable for previous volume sets only, and the purchaser will incur the UPS COD charge of $1.15. Personal checks, MC, VISA, American Express credit cards, and cash are acceptable means of payment. Checks drawn on foreign banks should include an amount sufficient to cover the current currency exchange rate.

ADVERTISING

Ad rate sheets will be sent to interested persons upon request to THE PAPER, P O Box 1142, Columbia MD 21044.

DEALERS

Dealers may order complete sets of the previous volumes as well as a minimum of five (5) copies of each month's current issue. Dealers are invited to inquire by mail or by telephone to (301) 730-5186

SOFTWARE

Software written for and distributed by THE PAPER is intended for use on the 8K PET, and we do not make any claims that said software is appropriate for use on any other Commodore computer system.

EDITORIAL

Hi there! I'm Sandy and I'm one of the newer additions to the Aresco staff. I started working here in August of '79 and I enjoy working here with Rick and Terry very much. Some of you may know of me already from talking to me over the phone or by mail. I used to type the newsletters but now I do the shipping of your subscriptions. Another addition to our staff is Bob Brock. You won't be hearing too much from him as he deals mostly with the financial side of Aresco. With Bob, this brings the total number of staff to four.

Many of you may be wondering why the Paper wasn't published in March. Well, that's because three out of the four of us have been out of work because of illness. Rick has been running Computer Crossroads with a kleenex in one hand and a bottle of Neo-Synephrine in the other. Terry has been at home in bed (calling at least three times a day) with some type of flu or virus that kept her there for three weeks. The absence of a newsletter is beginning to make sense, isn't it? Needless to say, we were unable to keep our deadline with the printer. This is why we decided to make a giant issue of the Paper for April. This is the March/April issue and it contains a lot of good stuff. We are trying to get back on schedule and we appreciate your patience with us. We are looking forward to all the articles you can send us too!

Terry- Back in the April issue (Vol 2, Iss 3, pg. 8) Roy Busdiecker expressed an opinion that the Paper ought to offer table of the 6502 instruction set by mnemonic and by opcode. Since I have not seen such a list published, I am offering you the lists that we have made up. We certainly found them useful for doing hand assembly of short routines or for disassembling routines published in the Paper as DATA statements with decimal opcodes. -Fran Turco

SEE NEXT PAGE FOR TABLES

6502 INSTRUCTION SET BY MNEMONIC

MNEMONIC	DECIMAL	HEX	OPERATION
ADC-ABS	109	6D	Add Memory to Accum with Carry
ADC-ABS,X	125	7D	
ADC-ABS,Y	121	79	
ADC-IMM	105	69	
ADC-IND,X	97	61	
ADC-IND,Y	113	71	
ADC-Z PAGE	101	65	
ADC-Z PAGE,X	117	75	
AND-ABS	45	2D	"AND" Accumulator with Memory
AND-ABS,X	61	3D	
AND-ABS,Y	57	39	
AND-IMM	41	29	
AND-IND,X	33	21	
AND-IND,Y	49	31	
AND-Z PAGE	37	25	
AND-Z PAGE,X	53	35	
ASL-A	10	0A	Shift Left One Bit (Accum)
ASL-ABS	14	0E	Shift Left One Bit (Mem)
ASL-ABS,X	30	1E	
ASL-Z PAGE	6	06	
ASL-Z PAGE,X	22	16	
BCC	144	90	Branch on Carry Clear
BCS	176	B0	Branch on Carry Set
BEQ	240	F0	Branch If Equal (Result = 0)

MNEMONIC	DECIMAL	HEX	OPERATION
BIT-ABS	44	2C	Test Bits in Accum with Memory
BIT-Z PAGE	36	24	
BMI	48	30	Branch on Minus
BNE	208	D0	Branch if Not Equal (Result ≠ 0)
BPL	16	10	Branch on Plus
BRK	0	00	Force Break
BVC	80	50	Branch on Overflow Clear
BVS	112	70	Branch on Overflow Set
CLC	24	18	Clear Carry Flag
CLD	216	D8	Clear Decimal Mode
CLI	88	58	Clear Interrupt Disable Bit
CLV	184	B8	Clear Overflow Flag
CMP-ABS	205	CD	Compare Accum with Memory
CMP-ABS,X	221	DD	
CMP-ABS,Y	217	D9	
CMP-IMM	201	C9	
CMP-IND,X	193	C1	
CMP-IND,Y	209	D1	
CMP-Z PAGE	197	C5	
CMP-Z PAGE,X	213	D5	
CPX-ABS	236	EC	Compare Index X with Memory
CPX-IMM	224	E0	
CPX-Z PAGE	228	E4	
CPY-ABS	204	CC	Compare Index Y with Memory
CPY-IMM	192	C0	
CPY-Z PAGE	196	C4	

MNEMONIC	DECIMAL	HEX	OPERATION
DEC-ABS	206	CE	Decrement Memory by one
DEC-ABS,X	222	DE	
DEC-Z PAGE	198	C6	
DEC-Z PAGE,X	214	D6	
DEX	202	CA	Decrement Index X by one
DEY	136	88	Decrement Index Y by one
EOR-ABS	77	4D	"Exclusive OR" Accum with Memory
EOR-ABS,X	93	5D	
EOR-ABS,Y	89	59	
EOR-IMM	73	49	
EOR-IND,X	65	41	
EOR-IND,Y	81	51	
EOR-Z PAGE	69	45	
EOR-Z PAGE,X	85	55	
INC-ABS	238	EE	Incr Memory by one
INC-ABS,X	254	FE	
INC-Z PAGE	230	E6	
INC-Z PAGE,X	246	F6	
INX	232	E8	Increment Index X by One
INY	200	C8	Increment Index Y by one
JMP-ABS	76	4C	Jump
JMP-IND	108	6C	
JSR	32	20	Jump to Subroutine
LDA-ABS	173	AD	Load Accum from Memory
LDA-ABS,X	189	BD	
LDA-ABS,Y	185	B9	
LDA-IMM	169	A9	

MNEMONIC	DECIMAL	HEX	OPERATION
LDA-IND,X	161	A1	Load Accum from Memory
LDA-IND,Y	177	B1	
LDA-Z PAGE	165	A5	
LDA-Z PAGE,X	181	B5	
LDX-ABS	174	AE	Load Index X from Memory
LDX-ABS,Y	190	BE	
LDX-IMM	162	A2	
LDX-Z PAGE	166	A6	
LDX-Z PAGE,Y	182	B6	
LDY-ABS	172	AC	Load Index Y from Memory
LDY-ABS,X	188	BC	
LDY-IMM	160	A0	
LDY-Z PAGE	164	A4	
LDY-Z PAGE,X	180	B4	
LSR-A	74	4A	Shift One Bit Right (Accum)
LSR-ABS	78	4E	Shift One Bit Right (Mem)
LSR-ABS,X	94	5E	
LSR-Z PAGE	70	46	
LSR-Z PAGE,X	86	56	
NOP	234	EA	No Operation
ORA-ABS	13	0D	"OR" Accum with Memory
ORA-ABS,X	29	1D	
ORA-ABS,Y	25	19	
ORA-IMM	9	09	
ORA-IND,X	1	01	
ORA-IND,Y	17	11	
ORA-Z PAGE	5	05	
ORA-Z PAGE,X	21	15	

MNEMONIC	DECIMAL	HEX	OPERATION
PHA	72	48	Push Accum onto Stack
PHP	8	08	Push Processor Status on Stack
PLA	104	68	Pull Accumulator from Stack
PLP	40	28	Pull Processor Status from Stack
ROL-A	42	2A	Rotate One Bit Left (Accum)
ROL-ABS	46	2E	Rotate One Bit Left (Mem)
ROL-ABS,X	62	3E	
ROL-Z PAGE	38	26	
ROL-Z PAGE,X	54	36	
RTI	64	40	Return from Interrupt
RTS	96	60	Return from Subroutine
SBC-ABS	237	ED	Subt Memory from Accum with Borrow
SBC-ABS,X	253	FD	
SBC-ABS,Y	249	F9	
SBC-IMM	233	E9	
SBC-IND,X	225	E1	
SBC-IND,Y	241	F1	
SBC-Z PAGE	229	E5	
SBC-Z PAGE,X	245	F5	
SEC	56	38	Set Carry Flag
SED	248	F8	Set Decimal Mode
SEI	120	78	Set Interrupt Disable Status
STA-ABS	141	8D	Store Accum in Memory
STA-ABS,X	157	9D	
STA-ABS,Y	153	99	
STA-IND,X	129	81	
STA-IND,Y	145	91	
STA-Z PAGE	133	85	

MNEMONIC	DECIMAL	HEX	OPERATION
STA-Z PAGE,X	149	95	
STX-ABS	142	8E	Store Index X in Memory
STX-Z PAGE	134	86	
STX-Z PAGE,Y	150	96	
STY-ABS	140	8C	Store Index Y in Memory
STY-Z PAGE	132	84	
STY-Z PAGE,X	148	94	
TAX	170	AA	Transfer Accum to Index X
TAY	168	A8	Transfer Accum to Index Y
TSX	186	BA	Transfer Stack Pointer to Index X
TXA	138	8A	Transfer Index X to Accum
TXS	154	9A	Transfer Index X to Stack Pointer
TYA	152	98	Transfer Index Y to Accum

IMM - Immediate Addressing - The operand is contained in the second byte of the instruction.

ABS - Absolute Addressing - The second byte of the instruction contains the 8 low order bits of the effective address. The third byte contains the 8 high order bits of the effective address.

Z PAGE - Zero Page Addressing - Second byte contains the 8 low order bits of the effective address. The 8 high order bits are zero.

A - Accumulator - One byte instruction operating on the accumulator.

Z PAGE,X Z PAGE,Y - Zero Page Indexed - The second byte of the instruction is added to the index (carry is dropped) to form the low order byte of the EA. The high order byte of the EA is zeros.

ABS,X-ABS,Y Absolute Indexed - The effective address is formed by adding the index to the second and third byte of the instruction.

(IND,X) - Indexed Indirect - The second byte of the instruction is added to the X index discarding the carry. The results points to a location on page zero which contains the 8 low order bits of the EA. The next byte contains the 8 high order bits.

(IND,Y) - Indirect Indexed - The second byte of the instruction points to a location in page zero. The contents of this memory location is added to the Y index. The result being the low order eight bits of the EA. The carry from this operation is added to the contents of the next page zero location. The result being the 8 high order bits of the EA.

6502 INSTRUCTIONS BY DECIMAL OP CODE

DECIMAL	HEX	MNEMONIC	OPERATION
0	00	BRK	Force Break
1	01	ORA-IND,X	"OR" Accum with Memory
5	05	ORA-Z PAGE	
6	06	ASL-Z PAGE	Shift Left One Bit (Mem)
8	08	PHP	Push Processor Status on Stack
9	09	ORA-IMM	
10	0A	ASL-A	Shift Left One Bit (Accum)
13	0D	ORA-ABS	
14	0E	ASL-ABS	
16	10	BPL	Branch on Plus
17	11	ORA-IND,Y	
21	15	ORA-Z PAGE,X	
22	16	ASL-Z PAGE,X	
24	18	CLC	Clear Carry Flag
25	19	ORA-ABS,Y	
29	1D	ORA-ABS,X	
30	1E	ASL-ABS,X	
32	20	JSR	Jump to Subroutine
33	21	AND-IND,X	"AND" Accumulator with Memory
36	24	BIT-Z PAGE	Test Bits in Accum with Memory
37	25	AND-Z PAGE	
38	26	ROL-Z PAGE	Rotate One Bit Left (Mem)
40	28	PLP	Pull Processor Status from Stack
41	29	AND-IMM	
42	2A	ROL-A	Rotate One Bit Left (Accum)
44	2C	BIT-ABS	
45	2D	AND-ABS	
46	2E	ROL-ABS	
48	30	BMI	Branch on Minus
49	31	AND-IND,Y	
53	35	AND-Z PAGE,X	
54	36	ROL-Z PAGE,X	
56	38	SEC	Set Carry Flag

DECIMAL	HEX	MNEMONIC	OPERATION
57	39	AND-ABS,Y	
61	3D	AND-ABS,X	
62	3E	ROL-ABS,X	
64	40	RTI	Return from Interrupt
65	41	EOR-IND,X	"Exclusive Or" Accum with Memory
69	45	EOR-Z PAGE	
70	46	LSR-Z PAGE	Shift One Bit Right (Mem)
72	48	PHA	Push Accum onto Stack
73	49	EOR-IMM	
74	4A	LSR-A	Shift One Bit Right (Accum)
76	4C	JMP-ABS	Jump
77	4D	EOR-ABS	
78	4E	LSR-ABS	
80	50	BVC	Branch on Overflow Clear
81	51	EOR-IND,Y	
85	55	EOR-Z PAGE,X	
86	56	LSR-Z PAGE,X	
88	58	CLI	Clear Interrupt Disable Bit
89	59	EOR-ABS,Y	
93	5D	EOR-ABS,X	
94	5E	LSR-ABS,X	
96	60	RTS	Return from Subroutine
97	61	ADC-IND,X	Add Memory to Accum with Carry
101	65	ADC-Z PAGE	
104	68	PLA	Pull Accumulator from Stack
105	69	ADC-IMM	
108	6C	JMP-IND	
109	6D	ADC-ABS	
112	70	BVS	Branch on Overflow Set
113	71	ADC-IND,Y	
117	75	ADC-Z PAGE,X	
120	78	SEI	Set Interrupt Disable Status
121	79	ADC-ABS,Y	
125	7D	ADC-ABS,X	
129	81	STA-IND,X	Store Accum in Memory

DECIMAL	HEX	MNEMONIC	OPERATION
132	84	STY-Z PAGE	Store Index Y in Memory
133	85	STA-Z PAGE	
134	86	STX-Z PAGE	Store Index X in Memory
136	88	DEY	Decrement Index Y
138	8A	TXA	Transfer Index X to Accum
140	8C	STY-ABS	
141	8D	STA-ABS	
142	8E	STX-ABS	
144	90	BCC	Branch on Carry Clear
145	91	STA-IND,Y	
148	94	STY-Z PAGE,X	
149	95	STA-Z PAGE,X	
150	96	STX-Z PAGE,Y	
152	98	TYA	Transfer Index Y to Accum
153	99	STA-ABS,Y	
154	9A	TXS	Transfer Index X to Stack Pointer
157	9D	STA-ABS,X	
160	A0	LDY-IMM	Load Index Y from Memory
161	A1	LDA-IND,X	Load Accum from Memory
162	A2	LDX-IMM	Load Index X from Memory
164	A4	LDY-Z PAGE	
165	A5	LDA-Z PAGE	
166	A6	LDX-Z PAGE	
168	A8	TAY	Transfer Accum to Index Y
169	A9	LDA-IMM	
170	AA	TAX	Transfer Accum to Index X
172	AC	LDY-ABS	
173	AD	LDA-ABS	
174	AE	LDX-ABX	
176	B0	BCS	Branch on Carry Set
171	B1	LDA-IND,Y	
180	B4	LDY-Z PAGE,X	
181	B5	LDA-Z PAGE,X	
182	B6	LDX-Z PAGE,Y	
184	B8	CLV	Clear Overflow Flag
185	B9	LDA-ABS,Y	

DECIMAL	HEX	MNEMONIC	OPERATION
186	BA	TSX	Transfer Stack Pointer to Index X
188	BC	LDY-ABS,X	
189	BD	LDA-ABS,X	
190	BE	LDX-ABS,Y	
192	C0	CPY-IMM	Compare Index Y with Memory
193	C1	CMP-IND,X	Compare Accum with Memory
196	C4	CPY-Z PAGE	
197	C5	CMP-Z PAGE	
198	C6	DEC-Z PAGE	Decrement Memory by one
200	C8	INY	Increment Index Y by one
201	C9	CMP-IMM	
202	CA	DEX	Decrement Index X by one
204	CC	CPY-ABS	
205	CD	CMP-ABS	
206	CE	DEC-ABS	Decrement Memory by one
208	D0	BNE	Branch if Not Equal (Result ≠ 0)
209	D1	CMP-IND,Y	
213	D5	CMP-Z PAGE,X	
214	D6	DEC-Z PAGE,X	
216	D8	CLD	Clear Decimal Mode
217	D9	CMP-ABS,Y	
221	DD	CMP-ABS,X	
222	DE	DEC-ABS,X	
224	E0	CPX-IMM	Compare Index X with Memory
225	E1	SBC-IND,X	Subt Memory from Accum with Borrow
228	E4	CPX-Z PAGE	
229	E5	SBC-Z PAGE	
230	E6	INC-Z PAGE	Increment Memory by 1
232	E8	INX	Increment Index X by 1
233	E9	SBC-IMM	
234	EA	NOP	No Operation
236	EC	CPX-ABS	
237	ED	SBC-ABS	
238	EE	INC-ABS	

DECIMAL	HEX	MNEMONIC	OPERATION
240	F0	BEQ	Branch If Equal (Result = 0)
241	F1	SBC-IND,Y	
245	F5	SBC-Z PAGE,X	
246	F6	INC-Z PAGE,X	
248	F8	SED	Set Decimal Mode
249	F9	SBC-ABS,Y	
253	FD	SBC-ABS,X	
254	FE	INC-ABS,X	

IMM - Immediate Addressing - The operand is contained in the second byte of the instruction.

ABS - Absolute Addressing - The second byte of the instruction contains the 8 low order bits of the effective address. The third byte contains the 8 high order bits of the effective address.

Z PAGE - Zero Page Addressing - Second byte contains the 8 low order bits of the effective address. The 8 high order bits are zero.

A - Accumulator - One byte instruction operating on the accumulator.

Z Page,X Z Page,Y - Zero Page Indexed - The second byte of the instruction is added to the index (carry is dropped) to form the low order byte of the EA. The high order byte of the EA is zeros.

ABS,X-ABS,Y Absolute Indexed - The effective address is formed by adding the index to the second and third byte of the instruction.

(IND, X) - Indexed Indirect - The second byte of the instruction is added to the X index discarding the carry. The results points to a location on page zero which contains the 8 low order bits of the EA. The next byte contains the 8 high order bits.

(IND, Y) - Indirect Indexed - The second byte of the instruction points to a location in page zero. The contents of this memory location is added to the Y index. The result being the low order eight bits of the EA. The carry from this operation is added to the contents of the next page zero location. The result being the 8 high order bits of the EA.

READER I/O

To: Dennis A Costarakis & Shawn Glisson - I finally managed to get smart enough to subscribe and dig up all the back issues of The PAPER! In the October '79 issue, the "SCREEN DUMP" routine caught my eye, so I had to crank up my unfinished "TEXT EDITOR" and give you the easy answer: MAKE A FAKE QUOTE!

Use the secondary address #5 and define a quote. Then just before the PRINT#9 statement, if SL=34, then SL=254. Simple, yes?

Now, why print a RVS or OFF before every letter? There must be a better way" What is the POKE 205,0 for? And why the CMD 9? Also, the whole thing needs different values for an old PET with new ROMs.

Has anyone found a way to make the 2023-2 LIST in lower case for instructions, etc.?

By the way...when writing subroutines with the BASIC TOOLKIT, I've found that if you refer to a subroutine by NAME instead of by number when you haven't got it in the program yet, the RENUMBER operation will not change it to 63999, and you can FIND it by name so much easier when it's time to put in its number. - R Vanderbilt Foster
****

People: Here are some notes about the software review by David Conley on "DUNJONQUEST". It is obvious that neither Mr Conley or the author of the game are very experienced at programming. The entire section on rolling dice in the review is proof enough. The problem is easily solved using the function capability of BASIC. Make one of the first executable statements of the program a DEF statement (and make sure it is only executed once). The statement is:

```
10 DEF NDR(X) = INT(RND(1) * X) +1
```

This causes a random integer from 1 to X to be generated on each call to FNR. To roll three dice, just invoke FNR three times:

```
2190 J=FNR(6)+FNR(6)+FNR(6)
2191 RETURN
```

The statement 20 J=RND(-TI) should be executed only once to seed the BASIC random number generator. This takes the place of the "RANDOMIZE" verb in other BASICs. - Jeff Pimper
****

Terry - I really enjoy The PAPER. Some of the items are a little over my head, some a lot over, and some I get a lot out of. But all in all, I enjoy it. I'd like to see some articles on programs for (or especially changing existing programs to run on) the CBM printer which I finally received 1.3 years after ordering. However, it does work, and I don't need more information than the revised manual offers. Keep sending out the good word...we need you! - Fred Minchin

Terry - James McArthur's "SEARCH" in Volume 2, issue 10, page 24, is an absolute GEM! I had decided not to renew, but have changed my mind, since I think this program alone is worth the year's subscription price. Sorry to say, I found little else of interest in Volume 2.

Of course, you didn't quite manage to type all the machine code correctly. In line 140, the 21st character in the string should be a 3 instead of a 2. I hope you'll print a note to this effect in your next issue; it would be regrettable if any readers missed out on this fine program because of a simple error. - J L Pietenpol
****

Terry - It was gratifying to see "SEARCH" published in the PAPER at last! Unfortunately, there are some errors which will keep it from working. About midway through line 140, there is the character sequence 9028. The 2 in this sequence should be a 3. Also, near the ends of lines 330 and 340, there should be two right parentheses together, instead of one. I noticed that the lines of code which were too long to fit in one line of print were continued on an additional line, which is ok, except that the additional line is shown spaced over from the left edge, which would result in a gap in the code. Each string of code within the quote marks has to be continuous.

The MERGE program which I mentioned in my last letter is ready, although it is perhaps somewhat outdated, considering some of the system hardware now on the market. This program has some advantages, at least; it doesn't require any alterations to the PET, and it doesn't cost anything (except several frustrating hours of typing and proofreading, since it is nearly three times as long as SEARCH). In any case, I will soon send along a copy of it, along with a listing made on a PET 2022 printer. - James F McArthur
****

Terry - I decided not to renew. I'm sorry, but the PAPER is too high level for me, and I don't get enough out of it. "Compute" is more to my taste. As noted by Roy Busdiecker, commenting on Compute vs. "Micro", Compute seems to be the more "customer" and less "tinkerer" oriented. The same is true of Compute vs. the PAPER. Also, your lack of advertising is positively detrimental, and indicated a purist and elitist attitude. I like advertising. I like to know what's available. Besides, the lack of advertising raises your prices to an unacceptable level, considering. - C A Cozart

> We've asked readers repeatedly how they feel about advertising - and the amount of ads we carry reflects the views of readers who responded. Sorry you won't be with us this year. We'll miss you and all the articles you've contributed... - Terry

****

Software Maintenance Bulletin

MICRO-SET I (for PET)

Version	Date	Comment

NOTE: Customers with versions prior to 1.73 may obtain a current version by sending their original cassette with $2.00 and their name and address to MSS.

1.73	9 Jun 79	Incorporates changes to make the program compatible with Version 2 PET's (Models 2001-16 and 2001-32), as well as the original Model 2001-8.
1.74	9 Oct 79	Discrepancy: Above changes caused malfunction in CREATE TAPE routine when used on program lines numbered 999 and below.

Correction: List line 6394Ø and position the cursor over the G of GOTO. Press the INST/DEL key 8 times. Move the cursor over into the special characters following GOTO 6396Ø, delete one of them, then press RETURN. List 6394Ø again, and compare it to the "Line after correction" shown below (the "Line containing error" is underlined to show portions to be deleted). Change version number and date in lines 15 and 6ØØØ7 as shown below.

```
LINE CONTAINING ERROR

 63940 PRINT"CMD1:LIST"T1:PRINT"CMD3:S="S":T1="T1":PV="PV":GOTO63960:TTTTT

LINE AFTER CORRECTION

 63940 PRINT"CMD1:LIST"T1:PRINT"CMD3:S="S":T1="T1":GOTO63960:TTTT";

               15 PRINT"|VERSION 1.74         9 OCT 79  |


               60007 REM  VERSION 1.74, 9 OCT 1979
```

NEW PRODUCT ANNOUNCEMENT

New Fantasy Software for
the PET, APPLE and TRS-80

Automated Simulations
P.O. Box 4232
Mountain View, CA 94040

Now Automated Simulations challenges PET, APPLE, and TRS-80 owners to rescue an entire city from the fireballs of Morloc the Mad.

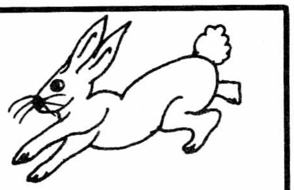
In Morloc's Tower the player must hunt through a maze of 30 rooms - all displayed on the screen - in search of the evil and elusive Morloc before the wisard can destroy the city of Hagedorn.

Morloc's Tower combines a challenging puzzle to solve with excellent graphics and 18 real-time command options. Dozens of frightening monsters of different shapes and sizes leap from the shadows to assault the player. Three kinds of rings, a magic sword, two amulets, and a half-a-dozen other treasures are hidden within the six-floor tower to aid - or hinder - the adventurer.

The competitive scoring system keeps the game challenging and exciting even after many of the tower's mysteries have been revealed. Three levels of play let the user adjust the difficulty of the game, while the Book of Lore not only explains the rules, but offers some helpful hints on solving the puzzles.

Morloc's Tower is designed for use on the Commodore PET (with at least 20K), the Radio Shack TRS-80 (Level II, 16K), and the APPLE II (32K with Applesoft in ROM).

Morloc's Tower is available at participating Byte Shops, Computerlands, and other dealers nation-wide, or for $14.95 from Automated Simulations.



THE END

by Roy Busdiecker

Not too long ago, while working on one of my oddball projects, I found that I needed a fast routine to find the end of whatever BASIC program happened to be in my PET. Obviously, the routine could not interfere with the BASIC program. A solution that would meet both needs (fast, and non-interfering) would be a machine language routine stored in the second cassette buffer. As a side benefit, that approach would also allow me to demonstrate the output of C.W. Moser's ASSM/TED (reviewed in Issue 6, Vol. 2).

Long-time readers of The Paper will recall a series of articles (Issue 9, Vol. 1: The PET Symbol Table and Data Formats; Issue 1, Vol. 2: A Decoder Add-On to the Mem-Explorer; Issue 2, Vol. 2: Exploring PET's Memory - A Real Program) describing the structure of BASIC programs in PET Basic. Summarizing briefly, the programs you type in start at location 1Ø25 in the PET's memory. The first two bytes hold the address of the beginning of the next program line (the number respresented by these two bytes is called a "link pointer"). The second pair of bytes hold the line number, following which are a variable number of bytes holding tokens and characters. The end of each line is marked with a Ø. Each line follows this same pattern, while the end of program is indicated by a link pointer whose value is zero.

Our strategy for the machine language routine is to "hopscotch" from one link pointer to the next, until we find one whose value is zero.

The program which accomplishes that strategy is shown in the listing produced by Mr. Moser's ASSM/TED. In that listing, the far left column gives the addresses (in hex) into which the machine language commands have been assembled. The next three columns are the hex values which correspond to the command mnemonics farther to the right. The next column holds the consecutive line numbers for the assembly source language which constitutes the rest of the listing. I typed in the labels, commands, addresses, and comments to the right, and the assembler converted them to the hex values that the 6502 microprocessor recognizes.

Now to a description of the program. The labels LO and HI, located at hex Ø33A and Ø33B (decimal 826,827) reserve space for the answer. After running the routine, a BASIC command PRINT PEEK (826) + 256 * PEEK (827) will retrieve the value.

The section from INIT (initialize) to LINKH sets hex Ø4Ø1 (decimal 1Ø25) as the address of the first link pointer, so the routine always starts looking in the right place.

The addresses following LDA in the LINKH line and the LINKL line are modified by the program as it runs, so they always hold the address of the next line pointer. The BEQ in the line following LINKH tests for an address of zero, and transfers control to the line labelled DONE when it find the zero. The section between LINKL and DONE puts the new addresses after the LDA's, and the JMP in the line before DONE transfers control back to LINKH to find the next link pointer.

Starting at the line labelled DONE, the program gets the address of the zero-value pointer, and stores (STA) its low byte in location LO (826), and its high byte in HI (827).

The RTS returns control to the routine that called it. If you enter the routine from BASIC with SYS 828, the RTS will return you to BASIC.

A one-line BASIC command that will run the machine language program and print out its result is shown below.

```
SYS 828:PRINT PEEK (826) + 256 * PEEK (827)
```

That's all there is to it!

If you don't have an Assembler program, the BASIC program below will load the appropriate values into the second cassette buffer. Once you have run it, you can load another BASIC program on top of it, and still have the machine language routine available via the SYS command (as long as you don't use the second cassette!).

```
6000 REM--- FIND-END ---
6001 REM BY ROY BUSDIECKER
6002 REM    MICRO SOFTWARE SYSTEMS
6003 REM    P.O. BOX 1442
6004 REM    WOODBRIDGE, VA  22193
6005 :
6006 REM    USE SYS828 TO RUN ML PROGRAM
6007 :
6008 REM    PRINT PEEK(826)+256*PEEK(827) FOR ANSWER
6009 FORI=1TO 62 :READK:POKE 825 +I,K:NEXTI
6010 DATA1,4,169,1,141,85,3,105,1,141,79,3,169,4,141,80,3,141,86,3,173,2,4
6020 DATA240,24,170,173,1,4,141,85,3,105,1,141,79,3,138,141,86,3,105,0,141
6030 DATA80,3,76,78,3,173,85,3,141,58,3,173,86,3,141,59,3,96

>
```

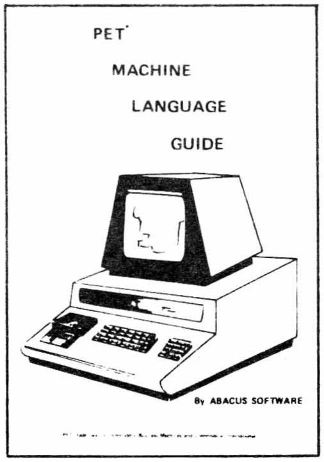
```
                0010 ;FIND-END - (C)1979, ROY BUSDIECKER
                0020 ;
                0030             .LS
                0040             .OS
                0050             .BA $033A
033A- 00        0060 LO          .BY 0
033B- 00        0070 HI          .BY 0
033C- A9 01     0080 INIT        LDA #01       ; INITIALIZE
033E- 8D 55 03  0090             STA LINKL+1   ; ADDR OF LO BYTE, NEXT PTR
0341- 69 01     0100             ADC #01
0343- 8D 4F 03  0110             STA LINKH+1   ; ADDR OF HI BYTE, NEXT PTR
0346- A9 04     0120             LDA #04       ; HI BYTE OF PTR ADDRESS
0348- 8D 50 03  0130             STA LINKH+2
034B- 8D 56 03  0140             STA LINKL+2
034E- AD 02 04  0150 LINKH       LDA $0402     ; GET HI BYTE OF
0351- F0 18     0160             BEQ DONE      ;  LINK POINTER
0353- AA        0170             TAX
0354- AD 01 04  0180 LINKL       LDA $0401     ; GET LO BYTE OF
0357- 8D 55 03  0190             STA LINKL+1   ;  LINK POINTER
035A- 69 01     0200             ADC #01
035C- 8D 4F 03  0210             STA LINKH+1
035F- 8A        0220             TXA
```

```
0360- 8D 56 03  0230       STA LINKL+2
0363- 69 00     0240       ADC #00
0365- 8D 50 03  0250       STA LINKH+2
0368- 4C 4E 03  0260       JMP LINKH
036B- AD 55 03  0270 DONE  LDA LINKL+1   ; PASS INFO TO
036E- 8D 3A 03  0280       STA LO        ;  BASIC
0371- AD 56 03  0290       LDA LINKL+2
0374- 8D 3B 03  0300       STA HI
0377- 60        0310       RTS
                0320       .EN

LABEL FILE:  [ / = EXTERNAL ]


LO=033A                HI=033B                INIT=033C
LINKH=034E             LINKL=0354             DONE=036B

//0000,0378,0378
```



TRY METAPRINTING
by John Matarella

Human cultures depend upon tradition to maintain a necessary stability in order to more effectively withstand the forces of change. Thus we carry old concepts and procedures over from the old media to the new, even when they are no longer necessary, or even desirable.

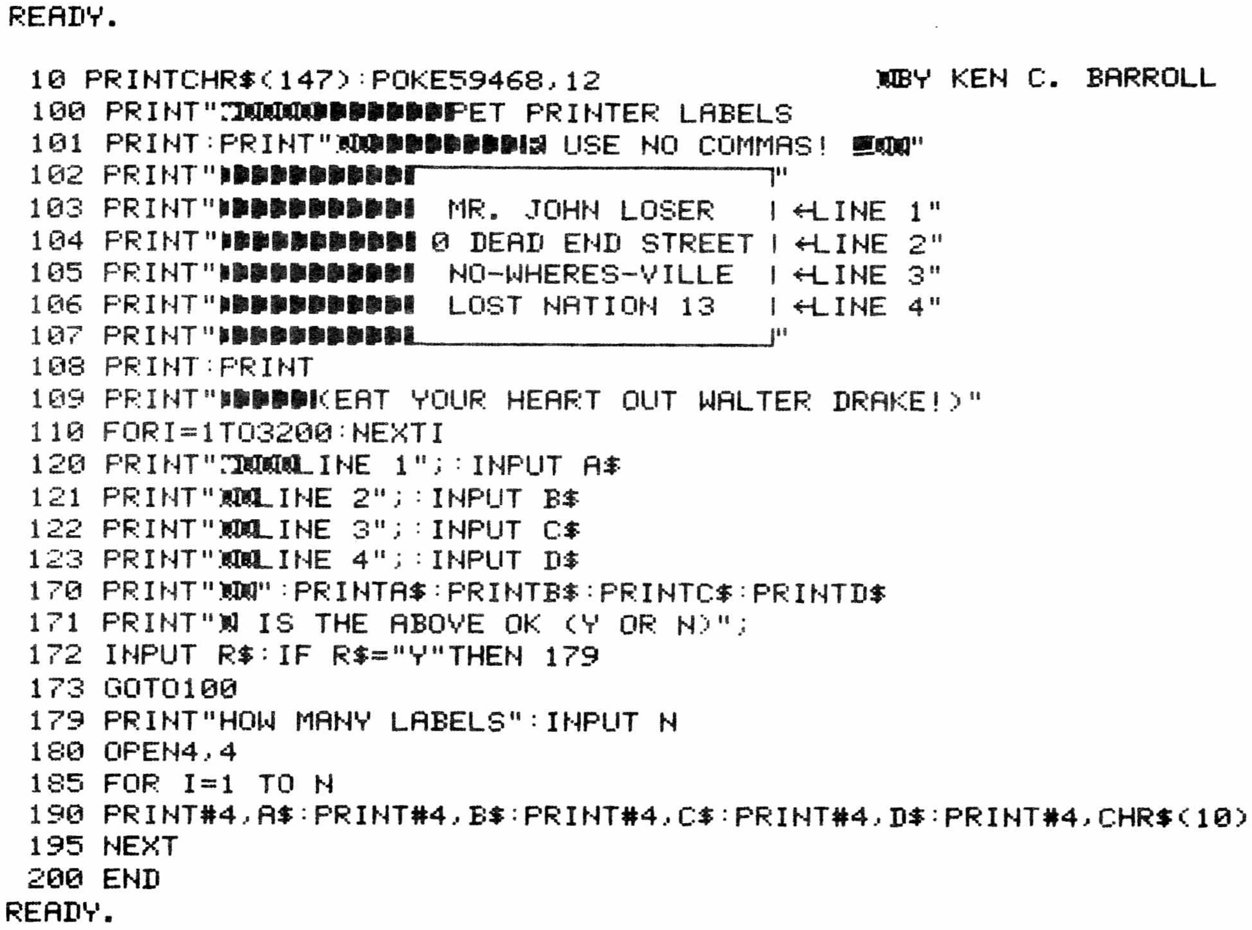
A perhaps trivial example of such an anachronism is the way we "write" on CRT's as if we were still dependent upon the mechanics of hard copy printing or writing.

Try the following METAPRINTING routine in the instruction section of some of your programs. If the effect is not eye-pleasing, it should at least be novel!

```
100 REM         METAPRINTING  -  SAMPLE PROGRAM
110 REM  TO KEEP IT SIMPLE WE WILL LIMIT ALL STRINGS TO LESS THAN 41
         CHARACTERS
120 REM  IN YOUR PROGRAM LET P$ EQUAL THE STRING TO BE PRINTED- THEN
         CALL THE SUBROUTINE
130 REM
140 REM
200 READ A$
210 IF A$="END" THEN POKE 513,0:WAIT 513,10:?CHR$(147):RUN
220 P$=A$
230 GOSUB 60000
240 GOTO 200
250 REM
260 REM
500 DATA "METAPRINTING DEMO","----------------------------------------":
    REM 40 DASHES
510 DATA "NOW IS THE TIME FOR ALL GOOD MEN","TO COME"
520 DATA "TO THE AID OF THEIR PARTY","*******"
530 DATA "FOUR SCORE AND SEVEN YEARS AGO","OUR FOREFATHERS BROUGHT FORTH","
    UPON THIS CONTINENT"
540 DATA "A NEW NATION","CONCEIVED IN LIBERTY","AND DEDICATED TO THE
    PROPOSITION","THAT ALL MEN","ARE CREATED EQUAL"
550 DATA END
560 REM
570 REM
60000 REM       METAPRINTING SUBROUTINE
60010 REM       BY DR MATARELLA
60020 REM
60030 LX=LEN(P$)
60040 IF LX/2<>INT(LX/2) THEN P$=P$+CHR$(32):LX=LX+1
60050 FOR IX = 1 TO LX/2
60060 ?TAB(20-IX)MID$(P$,LX/2+1-IX,1)CHR$(145)
60070 ?TAB(19+IX)MID$(P$,LX/2+IX,1)CHR$(145)
60080 NEXT IX
60090 ?:?
60100 RETURN
```

A SIMPLE MAIL LABEL PROGRAM

by Ken C Barroll



```
READY.

 10 PRINTCHR$(147):POKE59468,12                    ▮BY KEN C. BARROLL
 100 PRINT"▮▮▮▮▮▮▮▮▮▮▮▮▮▮PET PRINTER LABELS
 101 PRINT:PRINT"▮▮▮▮▮▮▮▮▮▮▮▮ USE NO COMMAS! ▮▮▮"
 102 PRINT"▮▮▮▮▮▮▮▮▮▮▮┌───────────────────┐"
 103 PRINT"▮▮▮▮▮▮▮▮▮▮▮  MR. JOHN LOSER    │ ←LINE 1"
 104 PRINT"▮▮▮▮▮▮▮▮▮▮▮ 0 DEAD END STREET │ ←LINE 2"
 105 PRINT"▮▮▮▮▮▮▮▮▮▮▮  NO-WHERES-VILLE   │ ←LINE 3"
 106 PRINT"▮▮▮▮▮▮▮▮▮▮▮  LOST NATION 13    │ ←LINE 4"
 107 PRINT"▮▮▮▮▮▮▮▮▮▮▮└───────────────────┘"
 108 PRINT:PRINT
 109 PRINT"▮▮▮▮▮(EAT YOUR HEART OUT WALTER DRAKE!)"
 110 FORI=1TO3200:NEXTI
 120 PRINT"▮▮▮▮LINE 1";:INPUT A$
 121 PRINT"▮▮LINE 2";:INPUT B$
 122 PRINT"▮▮LINE 3";:INPUT C$
 123 PRINT"▮▮LINE 4";:INPUT D$
 170 PRINT"▮▮":PRINTA$:PRINTB$:PRINTC$:PRINTD$
 171 PRINT"▮ IS THE ABOVE OK (Y OR N)";
 172 INPUT R$:IF R$="Y"THEN 179
 173 GOTO100
 179 PRINT"HOW MANY LABELS":INPUT N
 180 OPEN4,4
 185 FOR I=1 TO N
 190 PRINT#4,A$:PRINT#4,B$:PRINT#4,C$:PRINT#4,D$:PRINT#4,CHR$(10)
 195 NEXT
 200 END
READY.
```

NEW PRODUCT ANNOUNCEMENT

Queue has published a second edition of its free Educational Software catalogue for APPLE, PET and TRS-80 computers from major publishers. This caltalogue has greatly expanded listings and includes simulations and strategy games. All software can be ordered directly from Queue.

For further information, contact Monica Kantrowitz, President, QUEUE, 5 Chapel Hill Drive, Fairfield, CT 06432, or call (203) 372-6761.

A SCREEN PRINT MACHINE LANGUAGE PROGRAM

by Paul W Sparks

In an earlier article (THE PAPER Volume II, issue 3, August 1979, page 3) I described the use of a Sourthwest Technical Products Corporation printer (SWTPC PR-40) interfaced with a PET 2001 microcomputer via the user port. The primary purpose of selecting the PR-40 is that it is quite inexpensive ($250); the primary disadvantage with the system described is that it is quite slow. It was suggested that a machine language version of this program whould be written, and soon after that article was published, I received a letter from C L Buchanan of Camp Springs, Md. Mr. Buchanan indicated not only that he had read the article and subsequently purchased a PR-40, but he also provided a machine language version of the program emulating each step of the BASIC program. A slight refinement of that program is presented here.

The approach that will be taken in this article is to go into sufficient detail to lead the beginning machine (assembly) language programmer through the logic of the program in general, the logic of the user port registers in detail, and the assembly language programming steps required to accomplish this logic. If you aren't interested in assembly language programming, but would like to utilize the program, just copy the BASIC program used to POKE in the machine language steps (Listing 1), call it up with a SYS 841, and print away. On the other hand, if you are an advanced assembly language programmer, and don't care to go through the detailed description, go straight to listing 2 (do not pass GO, do not collect $200) and do with it as you will. Now, if there is anyone left, we will begin.

The BASIC program used is listed below. One major difference between this version and that presented in the previous article is that this version does have full handshake between the printer and the PET, whereas the earlier version used a one second delay to insure that the line had been printed. The other difference is that there were two mistakes in the previous article. In line 20100, an "I" was put in place of a "1", so it should read:

```
20100  FOR J=1 TO A1
```

Of a more serious nature, the most important step, POKEing the character into the user port, was omitted in my draft. Therefore you must add line 20145:

```
20145 POKE 59457,D2(I)
```

Now, with that out of the way, let's review the steps:

21000 POKE 59459,255	This sets the direction register for the User Port to all 1's to set the data port to output
21005 POKE 59468,PEEK(59468) OR 1	Sets the CA1 polarity for receive on the character accept.
21010 X=32768:K=0:A=0:A1=25	Initializes all constants
21020 Y=PEEK(X	Gets a character from the screen memory
21025 IF Y=28 THEN RETURN	This line is a method to terminate the print process. Any value corresponding to a character of the user's choice could be used. For example, Y =28 corresponds to the backslash ("\") being used as a delimiter.
21030 IF Y< 32 THEN Y=Y+64	This partially converts PET memory code to ASCII code.
21035 IF Y>128 THEN Y=Y-128	This line converts a reverse character.
21040 POKE 59457,Y	Place the character on the data register
21050 POKE 59468,PEEK(59468) AND 31 OR 192	This lets the data-ready handshake (CB2) to 0 (low).
21060 POKE 59468,PEEK(59468) OR 224	Sets the data-ready handshake to 1 (high) and the printer will accept the data.
21065 IF (PEEK(59468) AND 2) THEN 21070	Test to see if the data received handshake handshake has been sent by the printer. If so, then continue.
21068 GOTO 21065	If not, then look again.
21070 K=K+1: IF K<40 THEN 21020	Step to the next position. If this position is not 40 then go back and repeat the process.
21080 A=A+1:K=0:X=X+40	Step to do the next line. Zero the character index.
21100 GOTO 21020	Start over on a new line.

Let's review the logic used in lines 21005, 21050, 21060, and 21065. The AND logic will only be true (have a value of 1) if both A and B are true (have a value of 1). The OR (inclusive OR) logic is true (1) if either A or B are true. For instance, if you wanted to turn a bit position high, just OR that position with a 1 and it will be a 1. Alternatively, you can turn that bit position to 0 by ANDing that position with a 0. If you want to keep the original value, OR the bit position with a 1. The last basic principle to keep in mind is that if you perform a logic operation with a byte, it is done bit by bit. For example, 8 AND 127 = 8:

```
  8 = 00001000
127 = 01111111
```

ANDing these two numbers bit by bit will get a match of ones only in the "8" position. 8 OR 127 would be 127 because all bit positions except 128 (the leftmost bit) have a one in at least one of the numbers. If you were to AND any number with 255 (11111111), the original number would result (A AND 255=A).

Now what does all that have to do with the logic steps in this program? The whole purpose of these steps is to set up the registers and handshakes for the printer operation. Commodore PET USERS CLUB NEWSLETTER (Volume 1, issue 3) has a very fine table of the PIA and VIA memory and function locations bit by bit. The article was written by Karl Hildon. Although it is not obviously true, I think the table was supplied by Jim Butterfield. In any event, we will be concerned with memory locations 59459, 59458, 59468, and 59469. Location 59459 sets the direction of the User Port so if you POKE 255 (all ones) into that location, all lines are output. If any lines had been zeros, then those lines would have been input. Location 59457 is the data port, so we POKE the ASCII value of the character we want to print into that location.

Position 0 (the rightmost bit) of location 59468 is used to set up CA1 IN polarity. If we OR the value in that location with one (00000001), all the other bits will stay the same and bit zero will become a one if it isn't already one. We forced bit zero high to set the polarity without affecting the other bit values. The new value is then POKEd back into that memory location in line 21005. Lines 21050 and 21060 are similar. The leftmost three bits of memory location 59468 are the CB2 IN/OUT control elements. ANDing 31 (00011111) with the contents of location 59468 keeps the righthand five bits the same, and turns the lefthand three bits to zeros. Then this new value is ORed with 192 (11000000), which will turn the leftmost two bits to ones and keep all the other values unchanged. The result is 110xxxxx (where x is the original value). That value is POKEd back into 59468. Then we set bit five high by ORing 224 (11100000) with the value now in 59468. You should also satisfy yourself that ORing 32 would have been sufficient for our use. The new value is POKED back into 59468 and we have blinked bit 6 off and on, telling the printer that the data is on its way. The last logic operation is a WAIT operation and could be written as such. Bit 1 (2nd from the right) of location 59469 (Interrupt Flag Register) is the CA1 bit. The contents of 59469 are ANDed with 2 (00000010) resulting in a zero unless bit 1 is equal to one. Notice that we don't POKE this value back in; we just want to sample it. If bit one is 1, that means the printer has sent a "ready for data" signal and we are ready to proceed.

You may have noticed that I have gone to a lot of trouble to control just what bits in the control registers were changed. Why not just change the entire byte by POKEing 0 or 255 to turn it off or on? I think it is worth the trouble because the user port shares a VIA (Versatile Interface Adapter) with the internal controls of the PET. Even if you didn't have to be concerned with the hidden secrets of your computer, it could still become troublesome if you aren't careful. For example, playing around with byte position 59468 could alter the CB2 IN/OUT control, CB1 for cassette #2 control, shift from graphic to lower case (or vice versa), or alter CA1 IN polarity. I don't know about you, but I'm too chicken to play "register roulette", so I carefully change only those bits that need changing (and offer the same advice to you).

You should have an idea of the structure and logic of the algorithms that are used here. Table 2 is a listing of the assembly version of this program. It is placed in the second cassette buffer, but it could be placed in any other location with minor variations. You probably noticed that there are quite a few NOPs (non-active spacer or no operation) scattered throughout the program. You can attribute this to careless planning on my part (assuming I didn't know how long the program was going to be) or to careful forethought (allowing you the option and space to add improvements if you care to do so).

Let's go through the program, step by step, in its assembly version. The line number of the BASIC version that corresponds to the steps being described at the time will be followed by the specific assembly language steps and an explanation of their function.

```
841          NOP                 Just resting!
842,3        LDXIM 0             Put 0 in the X register
844,5        LDYIM 0             Put 0 in the Y register

**21000 POKE 59459,255

846,7        LDAIM 255           Put 255 in the accumulator
848,9,0      STA 59459           Put the contents of the accumu-
                                 lator into location 59459
```

Note: This is how you POKE.

```
** 21005 POKE 59468, PEEK(59468) OR 1
851,2,3    LDA 59468          Put the contents of location
                              59468 into the accumulator
```

Note: This is how you PEEK.

```
854,5      ORAIM 1            Inclusive OR the contents of
                              the accumulator with 1, and
                              place the result in the acc.

856,7,8    STA 59468          Place the contents of the
                              accumulator into location
                              59468.
```

```
** 21010 X=32768: K=0:A=0:A1=25
```

Note: The next few steps break up 32768 into high and low byte values bacause a byte can only contain a number up to 255. These values are placed into convenient memory locations for use later on.

```
859,0      LDAIM 128          Put 128 (high value) in the acc.
861,2,3    STA 1014           Place the contents of the acc.
                              into memory location 1014.
864,5      LDAIM 0            Put 0 (low value) in the acc.
866,7,8    STA 1013           Place the contents of the acc.
                              into memory location 1013.
```

Note: Now we are prepared to start the main part of the program.

```
**21020    Y=PEEK(X+
869,0,1    LDA 1013           Load the contents of location
                              1013 into the accumulator
872,3,4    STA 884            Put the contents of the acc.
                              into location 884
875,6,7    LDA 1014           Load the contents of location
                              1014 into the accumulator
878,9,0    STA 885            Put the contents of the acc.
                              into location 885
881        NOP                resting
882        NOP                resting
883,4,5    LDAX add+X         Load the contents of the address
                              location +X into the accumulator
```

Note: This is a utilization of absolute addressing with index. Any loading from an address is equivalent to PEEKing, whereas an immediate load just places a specific value in the accumulator.

(Editor's note: This is also an example of self-modifying code!)

```
** 21025 IF Y=28 THEN RETURN
886,7      CMPIM 28           Compare the contents of the acc.
                              with the value 28.
```

```
888,9       BNE 1                    If the Z flag = 0 (the compari-
                                     son proved the values equal)
                                     continue.  Otherwise, skip the
                                     next step.
```

Note: The positive value less than 128 following a branch is the number of steps that are passed over while going forward. That number plus one will be the address that is being branched to. If the number is greater than 127, then the branch is negative.

```
890         RTS                      Return from subroutine.  This
                                     will return the machine language
                                     program back to the operating
                                     system.
```

Note: If the Z flag<>0, instruction 890 will be passed over (skipped).

```
** 21035 IF Y>128 THEN Y=Y-128

891,2       CMPIM 128                Compare the accumulator with 128
893,4       BMI 3                    If the N flag =0 (the comparison
                                     is not negative) continue.  Other-
                                     wise, skip the next three steps.
895         SEC                      Set the carry bit to 1 (set up to
                                     subtract without carrying)
896,7       SBCIM 128                Subtract 128 from the contents of
                                     the accumulator and put the result
                                     into the accumulator.
```

Note: If the results were negative, steps 895,6, and 7 will be skipped. The program will go directily from step 893,4 to 898.

```
**21030 IF Y<32 THEN Y=Y+64

898,9       CMPIM 32                 Compare contents of acc with 32
900,1       BPL 2                    If the N flag =1 (comparison is
                                     negative) continue.  Otherwise,
                                     jump over the next two steps.
902,3       ADCIM 64                 Add 64 to the contents of the acc.
                                     and place the results in the acc.
```

Note: If the results were positive, step 900,1 would cause a skip over 902,3 and the next executed step would be 904,5,6.

```
** 21040 POKE 59457,Y

904,5,6     STA 59457                Place the contents of the acc.
                                     into location 59457
```

Note: We have now PEEKed a value from a screen position, converted it to ASCII code, and POKEd it into the user port.

```
** 21050 POKE 59468,PEEK(59468) AND 31 OR 192

907,8,9     LDA 59468                Place the contents of location
                                     59468 into the accumulator.
```

```
910,1       ANDIM             AND the contents of the acc.
                              with 31 and place the result
                              back into the accumulator
912,3       ORAIM 192         Inclusive OR the contents of
                              the acc. with 192 and put the
                              results back into the acc.
914,5,6     STA 59468         Put the contents of the acc into
                              location 59468 (set the hand-
                              shake low)
917         NOP               just resting.

** 21060 POKE 59468,PEEK(59468) OR 224

918,9,0     LDA 59468         Load the contents of location
                              59468 into the accumulator
921,2       ORAIM 224         Inclusive OR the contents of
                              the acc. with 224 and put the
                              results back into the acc.
923,4,5     STA 59468         Place the contents of the acc
                              into memory location 59468 (set
                              the handshake high).

** 21065 IF (PEEK(59469) AND 2) THEN 21070

926,7,8     LDA 59469         Plcae the contents of location
                              59469 into the accumulator
929,0       ANDIM 2           AND the contents of the acc.
                              with 2 and put the result back
                              into the accumulator.
931         NOP               resting
932         NOP               still resting
933,4       BEQ 247           If the Z flag = 0 (the comparison
                              is not equal) continue. Other-
                              wise, just eight steps back to
                              step 926.
```

Note: Branches backwards are often confusing. It turns out that one can branch + or - 128 using op codes. Values greater than 127 are backwards. To get the proper value, you must take the number of steps backward and subtract that number from 255. In this case, we need to branch from step 934 back to 926. So 934-926=8 and 255-8=247. Voila'!

```
** 21070 K=K+1: IF K<40 THEN 21020

935         INX               Increase the value in the X regis-
                              ter by one
936,7       CPXIM 40          Compare the contents of the X reg-
                              ister with 40.
938,9       BNE 185           If the Z flag = 0 (the contents
                              of the X register equals 40) then
                              continue. Otherwise, jump back
                              70 steps to step 869.
```

Note: This section checks to see if an entire line of 40 characters has been completed. If not, it returns to step 869 to get the next character on the line.

```
** 21000 A=A+1:K=0:X=X+40

940,1        LDXIM 0             Put 0 in the X register
942          INY                 Increase the contentes of the
                                 Y register by 1.

** 21090 IF A A1 THEN RETURN

943,4        CMPIM 25            Compare the contents of the Y
                                 register with 25 (A1).
945,6        BEQ 14              If the Z flag = 0 (the comparison
                                 is not equal) then continue.
                                 Otherwise jump over the next 14
                                 steps.
```

Note: We have now initialized for the next line and checked to see if we have completed the last line (25, since there are 25 lines on the screen). If not, continue.

```
**21080 A=A+1:K=0:X=X+40

947,8,9      LDA 1013            Load the contents of location 1013
                                 into the accumulator
950          CLC                 Clear the carry flag
951,2        ADCIM 40            Add 40 to the contents of the acc.
                                 and put the results back into it.
953,4,5      STA 1013            Place the contents of the acc.
                                 into location 1013
956,7        BCS 4               If the carry flag (C) = 0, then
                                 continue. Otherwise skip the
                                 next four steps.
```

Note: The reason for this branch is to add 1 to the high value if the low value has a carry after 40 has been added to it.

```
958,9,0      JMP 869             Jump back to location 869 for the
                                 next instruction.
961          RTS                 Return from subroutine
```

Note: The only way to get to this instruction is if the comparison in step 943 succeeds (the complete screen has been read).

```
962,3,4      INC 1014            Increase the value in memory
                                 location 1014 by 1 and put the
                                 result back into location 1014
```

Note: Increase the high value and...

```
965,6,7      JMP 869             Jump back to step 869 for the
                                 next instruction.
968          BRK                 Interrupt
```

Note: The interrupt here isn't necessary; it's just a good practice.

A program will normally have several elements or sections in it that are not portable. This means that information in these sections would have to be changed if the program were located in any part of memory other than the addresses specified in the listing. In this program there are three such areas, and if you want to put the program elsewhere (relocate it), you'll want to be aware of them.

First, locations 1013 and 1014 are used as registers to store the high and the low values for the start of the line address. These locations should be good on either the old or the new ROMs, and any other convenient location would do as well.

Second, the values for locations 884 and 885 are specified in steps 872 and 878. If you decide to relocate the program, the addresses specified in these steps would have to be changed.

Third, steps 958 and 965 specify a jump location (step 869). If your program is relocated, the target address in these steps would have to be changed. The remainder of the program is relative and therefore portable.

Some adventurous beginners may be interested in making minor changes to the program for their own specific uses. For example, I used a backslash as a delimiter (end of print symbol). The backslash value is 28 (ASC=92). You could just as easily use a "@" (ASC=64) or any other character that you like. Just remember to convert the ASCII code to PET code!

You might want to print out only a part of the screen. If, for instance, you don't want to print the top two screen lines, then you'll have to increase the values stored in locations 1013 and 1014 (steps 859 and 864). I used a value of 32768 - the value for the top left-hand corner of the screen, since I wanted the entire screen to be printed out. Or you might want to omit the last two lines of screen display from your printout. To do this, decrease the value of the constant (A1) in steps 943,4 to 23. Similar techniques (changing the values of the constants) can be used to print any part of the screen or to address various portions of it for graphics.

Suppose you have a 96 character printer - or a printer with all the PET characters. The programming corresponding to BASIC lines 21030 and 21035 would have to be expanded. It might be appropriate to include the new conversion rules after step 894 and move the rest of the program forward. If you do this, remember to change the number of steps to be jumped backwards over in order to get to step 869. And, if you're using a PR-40 printer, replace steps 895-897 with:

```
895    NOP
896,7  CMPIM 32
```

and get a space inserted any time a graphic symbol is encountered. Then you'll be able to write in the appropriate symbol after the printing is completed.

Now this is a fairly straightforward program. The need for the extensive detail has been demonstrated repeatedly by the fact that there are few useful books on the subject (most define the opcodes and the structure of the CPU and associated chips). More than 10 reference books and considerable effort was required to get this program together! I feel that there is a great need for tutorial articles such as this, presenting "cookbook"

LISTING 1 - THE BASIC PROGRAM

```
10 FOR I= 840 TO 968 : READ J : POKE I,J : NEXT I : END
20 REM ** TO USE, TYPE SYS(840) **
30 REM **
9100 DATA 234,234,162,0,160,0,169,255,141,67,232,173,76
9110 DATA 232,9,1,141,76,232,169,128,141,246,3,169,0,141
9120 DATA 245,3,173,245,3,141,116,3,173,246,3,141,117,3
9130 DATA 234,234,189,168,130,201,28,208,1,96,201,128,48
9140 DATA 3,56,233,128,201,32,16,2,105,64,141,65,232,173
9150 DATA 76,232,41,31,9,192,141,76,232,234,173,76,232,9
9160 DATA 224,141,76,232,173,77,232,41,2,234,234,240,247
9170 DATA 232,224,40,208,185,162,0,200,192,25,240,14,173
9180 DATA 245,3,24,105,40,141,245,3,176,4,76,101,3,96,238
9190 DATA 246,3,76,101,3,0
```

LISTING 2 - ASSEMBLY LANGUAGE

```
841 0349   EA         NOP
842 034A   A2 00      LDXIM  0
844 034C   A0 00      LDYIM  0
846 034E   A9 FF      LDAIM  255
848 0350   8D 43 E8 STA    59459
851 0353   AD 4C E8 LDA    59468
854 0356   09 01      ORAIM  1
856 0358   8D 4C E8 STA    59468
859 035B   A9 80      LDAIM  128
861 035D   8D F6 03 STA    1014
864 0360   A9 00      LDAIM  0
866 0362   8D F5 03 STA    1013
869 0365   AD F5 03 LDA    1013
872 0368   8D 74 03 STA    884
875 036B   AD F6 03 LDA    1014
878 036E   8D 75 03 STA    885
881 0371   EA         NOP
882 0372   EA         NOP
883 0373   BD C0 03 LDAX   33728
886 0376   C9 1C      CMPIM  28
888 0378   D0 01      BNE    1
890 037A   60         RTS
891 037B   C9 80      CMPIM  128
893 037D   30 03      BMI    3

895 037F   38         SEC
896 0380   E9 80      SBCIM  128
898 0382   C9 20      CMPIM  32
900 0384   10 02      BPL    2
```

```
902 0386   69 40      ADCIM  64
904 0388   8D 41 E8 STA    59457
907 038B   AD 4C E8 LDA    59468
910 038E   29 1F      ANDIM  31
912 0390   09 C0      ORAIM  192
914 0392   8D 4C E8 STA    59468
917 0395   EA         NOP
918 0396   AD 4C E8 LDA    59468
921 0399   09 E0      ORAIM  224
923 039B   8D 4C E8 STA    59468
926 039E   AD 4D E8 LDA    59469
929 03A1   29 02      ANDIM  2
931 03A3   EA         NOP
932 03A4   EA         NOP
933 03A5   F0 F7      BEQ    247
935 03A7   E8         INX
936 03A8   E0 28      CPXIM  40
938 03AA   D0 B9      BNE    185
940 03AC   A2 00      LDXIM  0
942 03AE   C8         INY

943 03AF   C0 19      CPYIM  25
945 03B1   F0 0E      BEQ    14
947 03B3   AD F5 03 LDA    1013
950 03B6   18         CLC
951 03B7   69 28      ADCIM  40
953 03B9   8D F5 03 STA    1013
956 03BC   B0 04      BCS    4
958 03BE   4C 65 03 JMP    869
961 03C1   60         RTS
962 03C2   EE F6 03 INC    1014
965 03C5   4C 65 03 JMP    869
968 03C8   00         BRK
```

OBSERVATIONS ON VOLUME 2, ISSUE 10
by Roy Busdiecker

The product review on the PET 2022 tractor feed printer was quite interesting to me, since I recently sold my PET 2023 friction feed printer and replaced it with a tractor feed model by Base 2 Inc (priced at $600!). From my point of view, the good points about the 2023 were its fairly reasonable price and the fact that it prints a reasonable facsimile of the PET character set. One bad feature was the lack of tractor feed (due to my unwillingness to put out another $150 to get the tractor feature). It wouldn't have bothered me if the friction feed had worked more evenly, but the paper would begin skewing one way or the other after two or three pages ... bothersome when listing a long program.

More serious on the negative side was the fact that the printer does not print what you see on the screen. That means that listings which include lower-case letters on the screen will have those letters printed as graphic characters when you list the program on the printer. It also means that progams designed for output to the printer must have different statements than ones which put their output on the screen. The decision was a disappointment to me ... especially in light of Tandy's decision to make the Radio Shack TRS-80 formats have the same result on screen or printer.

James McArthur's article, SEARCH, illustrated several important factors. Since the BASIC program printed with the article loads a machine-language (ML) program into the second cassette buffer, the ML program is not lost when you load a BASIC program or do a system reset with something like NEW-CURSOR or UNCRASHER (International Technical Systems, PO Box 264, Woodbridge, VA 22194). Therefor, if you load and run SEARCH, you can then load as many other programs as you like, using the SEARCH routine on each one in turn ... but don't turn off your PET, or you'll have to start over. Compare this capability to the FIND function described by Donald Sheward in his review of The Programmer's Tookit. This is a good example of the tradeoff of cost, time, and ease of use between software (SEARCH) and hardware or firmware (FIND) methods of accomplishing the same result. The Toolkit, I believe, costs $50 or $80 (depending which version PET you have), but is available almost instantly, once the chip has been installed. SEARCH is free, but must be loaded and run each time the PET is turned on, if it is to be used.

On my old PET (2001-4/8 with outboard memory), I once located some user memory (RAM) in the free address space above the video monitor memory, and loaded an improved system monitor and other utility routines in that area at the beginning of each session ... then if control were lost, I could do a system reset with NEW-CURSOR and have those routines available immediately (even though the BASIC program was lost).

Several other comments on SEARCH:

- Lines 300-380 provide a general routine for loading hex versions of machine language programs formatted as in lines 110-210.

- If you have a "version 2" (new) PET, or a "version 1" (old) PET with a monitor program, you can load the hex ML program between the quotes in lines 120-210 using the monitor ... this will save typing

typing the whole BASIC program. Start loading at location 033A (the hex equivalent of 826). You can also use the monitor to save the ML programs directly, as shown below.

```
     Old PETS:   .S 01,SEARCH,033A,0378
     New PETS:   .S "SEARCH",01,033A,0378
```

- It would be most helpful if designers of machine language programs would provide "assembly listings", like the one in the same issue at the bottom of page 16 on the "Machine Language Programming" article by Abacus Software. Without the assembly version, it's beyond the ability of many readers to figure out how the program "does its thing". Even for those who can figure it out, its a lot of work. For those who are interested, I'm enclosing a "disassembly listing" which shows all the ML commands ond operands, but does not have the valuable comments and names of variables which can be provided only by the programmer.

In my article, "PET User Notes Replaced by COMPUTE.", the name of the magazine was changed to COMPUTER in the first line ... the "R" is in error, the correct name is COMPUTE.

LATE RUMOR: Commodore is reported to have a new CBM computer in the works with a 25-line by 80 column display ... sounds like it's oriented to the Word Processor market ... also a Word-Pro 3 ... also a dual floppy drive with more than a megabyte of storage!

That's it for now. I'll be looking forward to the next issue. Happy PETing!

```
HEX LISTING OF SEARCH (JAMES MCARTHUR)
     BY ROY BUSDIECKER


       PC   IRQ   SR AC XR YR SP
 .;   0401 E62E 32 04 5E 00 F8
 .
 .:   0338 00 00 A2 18 20 6A C4 A2
 .:   0340 FF E8 A8 B5 22 D0 FA 86
 .:   0348 21 CA D0 05 C0 20 D0 01
 .:   0350 60 A2 01 A0 04 86 1A 84
 .:   0358 1B A0 03 B1 1A 99 1C 00
 .:   0360 88 10 F8 A5 1C D0 04 A5
 .:   0368 1D F0 CF A4 1E A5 1F 84
 .:   0370 B2 85 B1 A2 90 28 20 1B
 .:   0378 DB 20 B1 DC E8 BD FE 00
 .:   0380 00 A0 03 C8 E8 B1 1A 9D
 .:   0388 FE 00 C9 22 D0 02 E6 42
 .:   0390 A5 42 29 01 D0 29 B1 1A
 .:   0398 10 25 C9 CB B0 21 84 44
 .:   03A0 A0 00 C8 B9 90 C0 10 FA
 .:   03A8 DE FE 00 30 F5 CA E8 C8
 .:   03B0 B9 90 C0 29 7F 9D FE 00
 .:   03B8 B9 90 C0 10 F1 A4 44 E0
 .:   03C0 B0 B0 04 B1 1A D0 BC A9
 .:   03C8 0D 9D FE 00 86 45 A4 21
 .:   03D0 BD FE 00 D9 21 00 D0 11
 .:   03D8 CA 88 D0 F4 C8 B9 FE 00
 .:   03E0 20 D2 FF C9 0D D0 F5 F0
 .:   03E8 08 A4 21 C6 45 A6 45 D0
 .:   03F0 DF A6 1C A4 1D 4C 55 03
 .:   03F8 00 00 F7 E7 00 00 00 00
```

DISASSEMBLY LISTING OF SEARCH (JAMES MCARTHUR)
BY ROY BUSDIECKER

```
826    033A A218    LDX #18
828    033C 206AC4  JSR C46A
831    033F A2FF    LDX #FF
833    0341 E8      INX
834    0342 A8      TAY
835    0343 B522    LDA 22,X
837    0345 D0FA    BNE 0341
839    0347 8621    STX 21
841    0349 CA      DEX
842    034A D005    BNE 0351
844    034C C020    CPY #20
846    034E D001    BNE 0351
848    0350 60      RTS
849    0351 A201    LDX #01
851    0353 A004    LDY #04
853    0355 861A    STX 1A
855    0357 841B    STY 1B
857    0359 A003    LDY #03
859    035B B11A    LDA (1A),Y
861    035D 991C00  STA 001C,Y
864    0360 88      DEY
865    0361 10F8    BPL 035B
867    0363 A51C    LDA 1C
869    0365 D004    BNE 036B
871    0367 A51D    LDA 1D
873    0369 F0CF    BEQ 033A
875    036B A41E    LDY 1E
877    036D A51F    LDA 1F
879    036F 84B2    STY B2
881    0371 85B1    STA B1
883    0373 A290    LDX #90
885    0375 28      PLP
886    0376 201BDB  JSR DB1B
889    0379 20B1DC  JSR DCB1
892    037C E8      INX
893    037D BDFE00  LDA 00FE,X
896    0380 00      BRK
897    0381 A003    LDY #03
899    0383 C8      INY
900    0384 E8      INX
901    0385 B11A    LDA (1A),Y
903    0387 9DFE00  STA 00FE,X
906    038A C922    CMP #22
908    038C D002    BNE 0390
910    038E E642    INC 42
912    0390 A542    LDA 42
914    0392 2901    AND #01
916    0394 D029    BNE 03BF
```

```
918    0396 B11A    LDA (1A),Y
920    0398 1025    BPL 03BF
922    039A C9CB    CMP #CB
924    039C B021    BCS 03BF
926    039E 8444    STY 44
928    03A0 A000    LDY #00
930    03A2 C8      INY
931    03A3 B990C0  LDA C090,Y
934    03A6 10FA    BPL 03A2
936    03A8 DEFE00  DEC 00FE,X
939    03AB 30F5    BMI 03A2
941    03AD CA      DEX
942    03AE E8      INX
943    03AF C8      INY
944    03B0 B990C0  LDA C090,Y
947    03B3 297F    AND #7F
949    03B5 9DFE00  STA 00FE,X
952    03B8 B990C0  LDA C090,Y
955    03BB 10F1    BPL 03AE
957    03BD A444    LDY 44
959    03BF E0B0    CPX #B0
961    03C1 B004    BCS 03C7
963    03C3 B11A    LDA (1A),Y
965    03C5 D0BC    BNE 0383
967    03C7 A90D    LDA #0D
969    03C9 9DFE00  STA 00FE,X
972    03CC 8645    STX 45
974    03CE A421    LDY 21
976    03D0 BDFE00  LDA 00FE,X
979    03D3 D92100  CMP 0021,Y
982    03D6 D011    BNE 03E9
984    03D8 CA      DEX
985    03D9 88      DEY
986    03DA D0F4    BNE 03D0
988    03DC C8      INY
989    03DD B9FE00  LDA 00FE,Y
992    03E0 20D2FF  JSR FFD2
995    03E3 C90D    CMP #0D
997    03E5 D0F5    BNE 03DC
999    03E7 F008    BEQ 03F1
1001   03E9 A421    LDY 21
1003   03EB C645    DEC 45
1005   03ED A645    LDX 45
1007   03EF D0DF    BNE 03D0
1009   03F1 A61C    LDX 1C
1011   03F3 A41D    LDY 1D
1013   03F5 4C5503  JMP 0355
```

MEM EXPLORER - AGAIN

by Ray Davidson

Upon paying my initial subscription, I received seven issues of THE PAPER, which I found to contain a great deal of interesting information. One program particularly useful for me was the MEM EXPLORER by Roy Busdiecker. Having an extra memory expansion, I needed to check for the transition from static to dynamic memory, in order to avoid porblems with INPUT and DATA statements. The usefulness of the program was marred somewhat by having to merge it with another program, and I set about trying to reduce it to something less than a screenful. I needed to be able to load it and list it on the screen, then load a new program, then RETURN over each line of the MEM EXPLORER in order to merge the two programs.

Obviously, a machine language program was called for in order to eliminate many of the DATA statements. The result is shown in Program 1, and the coding and mnemonics are given.

It still seemed that there were too many DATA statements. In fact, there were two lines of DATA for a machine language program which was contained in half the memory as a single line of BASIC! I decided to try to include the machine code in a single line of BASIC, and the result of that effort is found in program 2. When the program is run, the first three lines of the original program can be deleted. Line 1 carries the whole machine program, which can be loaded and saved quite conveniently with the BASIC code. This method can be used for any short machine program needed by BASIC, or for any larger ML program which can be broken down into units of subroutines. The hash after the machine code can be extended to include the name of the program. If other routines are included, program 3 is useful for locating addresses. One word of warning: Do not RETURN over line 1 and expect the program to work. Owing to the differences between ASCII and PEEK and POKE values, you will have trouble with bit 6 and will knock 64 off some addresses.

"IF P*Q" (which is the same as IF (P<>Ø) AND (Q<>Ø) was left over from the development stages, and "IF Q" can replace it. The ML routine in program 1 is not the shortest or most efficient, but a shorter program wouldn't have made such a good example.

PROGRAM #1

```
60000 FOR I=826 TO 857 : READ C : POKE I,C : NEXT
60010 DATA 165,80,41,127,170,232,160,255,200,185,145,192,16,
     250,202,208
60020 DATA 247,200,185,145,192,41,127,32,210,255,217,145,192,
     240,242,96
60030 B$=CHR$(34)+CHR$(20):INPUT"A.";K
60040 Q=0:B=256+PRINT"/C/":FOR J=0 TO 19 : P=1:L=K+J:M=PEEK
     (L):PRINT L;TAB(8);M;TAB(15);
60050 IF (Q=0) AND M>127 AND M<202 THEN POKE 80,M:SYS1034:PRI
     NT"/U/":GOTO60090
```

```
60060 IF M=34 THEN Q=0↑Q:PRINT B$;
60070 IF P*Q THEN PRINT B$;CHR$(M);B$;"/U/":P=0
60080 IF P AND M<>98 THEN PRINT CHR$(M)
60090 S=PEEK(L+1):PRINT TAB(23);M+B*S;TAB(30);M*B+S:NEXT:PR
     INT:GOTO 60030
```

```
ADDRESS
DEC  HEX      OP CODES    MNEMONIC   OPERANDS

826 033A      A5 50       LDAZ       80
828 033C      29 7F       ANDIM      127
830 033E      AA          TAX
831 033F      E8          INX
832 0340      A0 FF       LDYIM      255
834 0342      C8          INY
835 0343      B9 91 C0    LDAY       49297
838 0346      10 FA       BPL        834
840 0348      CA          DEX
841 0349      D0 F7       BNE        834
843 034B      C8          INY
844 034C      B9 91 C0    LDAY       49297
847 034F      29 7F       ANDIM      127
849 0351      20 D2 FF    JSR        65490
852 0354      D9 91 C0    CMPY       49297
855 0357      F0 F2       BEQ        843
857 0359      60          RTS
```

#2

```
EADY.
    1 REM "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
60000 FOR I=1034 TO 1065:READ C:POKE I,C:NEXT
60010 DATA 165,80,41,127,170,232,160,255,200,185,145,192,16,250,202,208
60020 DATA 247,200,185,145,192,41,127,32,210,255,217,145,192,240,242,96
60030 B$=CHR$(34)+CHR$(20):INPUT "A.",K
60040 Q=0:B=256:PRINT "[CLR]":FOR J=0 TO 19:P=1:L=K+J:M=PEEK(L):PRINT L TAB(8)
      M TAB(15);
60050 IF(Q=0)AND M>127 AND M<202 THEN POKE 80,M SYS 1034:PRINT "[UP]":GOTO 600
      90
60060 IF M=34 THEN Q=0↑Q:PRINT B$;
60070 IF P*Q THEN PRINT B$ CHR$(M)B$"[UP]":P=0
60080 IF P AND M<>98 THEN PRINT CHR$(M),
60090 S=PEEK(L+1):PRINT TAB(23)M+B*S,TAB(30)M*B+S:NEXT:PRINT:GOTO 60030
```

#3

```
1 REM"▒▒▒▌P)? ▙ π ▙▀┣▙▘┬ ▙▀┣)? ─π ┌┣ ┬ ▒▒▒▒"
60030 B$=CHR$(34)+CHR$(20):INPUT"A.",K
60040 Q=0:B=256:PRINT"▜":FORJ=0TO19:P=1:L=K+J:M=PEEK(L):PRINTLTAB(8)MTAB(15);
60350 IF(Q=0)ANDM>127ANDM<202THENPOKE80,M SYS1034:PRINT"▜":GOTO60090
60060 IFM=34THENQ=0↑Q:PRINTB$;
60070 IFP*QTHENPRINTB$CHR$(M)B$"▜":P=0
60080 IFPANDM<>98THENPRINTCHR$(M);
60090 S=PEEK(L+1):PRINTTAB(23)M+B*S;TAB(30)M*B+S:NEXT:PRINT:GOTO60030
EADY.
```

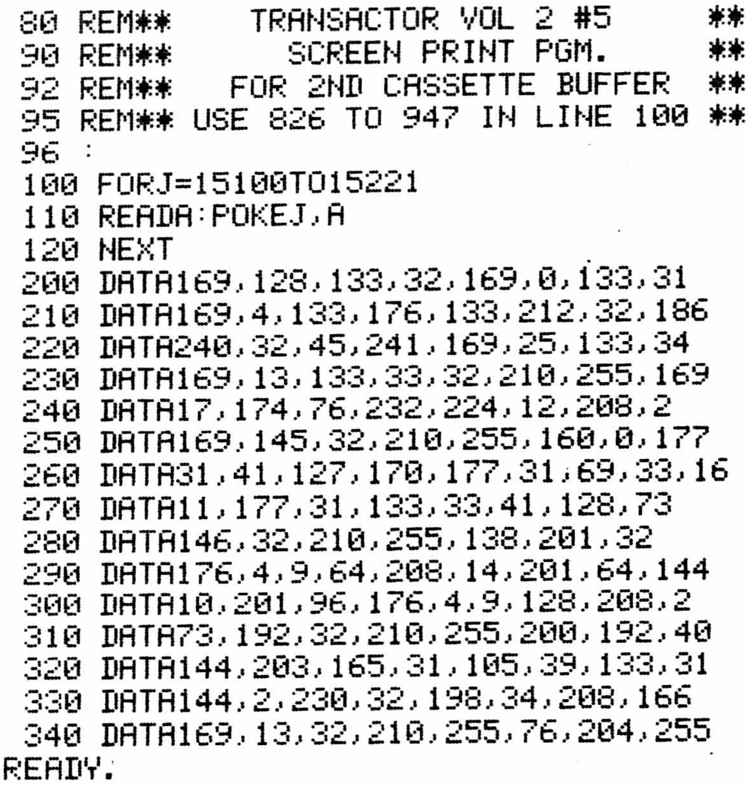
Here is a listing from Commodore's "The Transactor", Volume Two, Issue #5.

It is a basic load for a machine language pgm for screen to printer dump. It is short and easy to use and one of the most useful I have run across. I have found it best for listings that have both upper and lower case as it makes them readable without having to trnaslate the graphic characters. It's other handy use is for copies of instructions for new games or old games for new players whenever a quick reference is required.

It was originally intended to reside in the 2nd cassette buffer, but I found that if it was put in upper end of memory it didn't interfere with Toolkit (TM) which uses a portion of the buffer. I located it from 15100 to 15221 but for 8K you could use 7045 to 7166.

As it is quite short is can remain tucked away until called by SYS(XXXX) directly or as part of a basci program when it will direct to printer and then continue executing the program normally.

The program didn't say who originated it, but it is a useful utility program and my thanks to whoever it was.



```
 80 REM**    TRANSACTOR VOL 2 #5      **
 90 REM**       SCREEN PRINT PGM.     **
 92 REM**   FOR 2ND CASSETTE BUFFER  **
 95 REM** USE 826 TO 947 IN LINE 100 **
 96 :
 100 FORJ=15100TO15221
 110 READA:POKEJ,A
 120 NEXT
 200 DATA169,128,133,32,169,0,133,31
 210 DATA169,4,133,176,133,212,32,186
 220 DATA240,32,45,241,169,25,133,34
 230 DATA169,13,133,33,32,210,255,169
 240 DATA17,174,76,232,224,12,208,2
 250 DATA169,145,32,210,255,160,0,177
 260 DATA31,41,127,170,177,31,69,33,16
 270 DATA11,177,31,133,33,41,128,73
 280 DATA146,32,210,255,138,201,32
 290 DATA176,4,9,64,208,14,201,64,144
 300 DATA10,201,96,176,4,9,128,208,2
 310 DATA73,192,32,210,255,200,192,40
 320 DATA144,203,165,31,105,39,133,31
 330 DATA144,2,230,32,198,34,208,166
 340 DATA169,13,32,210,255,76,204,255
READY.
```

-D. Sheward

PET USERS

Glenn Schwartz
807 Avon
Philadelphia, PA 19116

John Loofbourrow
ACGNJ
(201) 233-7068

United PET Users
1929 Northport Dr. No. 6
Madison, WI 53704

Twin Cities PET Users
(John Fung)
(612) 376-5465

John Jones
2134 NE 45th Avenue
Portland, OR 97213

Sacramento PET Workshop
P.O. Box 28314
Sacramento, CA

Midpeninsula PUG
Ford Aerospace Cafeteria
3939 Fabian Way
Palo Alto, CA
(415) 328-7745 (Harry Saal)

BAMUG
1450 53rd Street
Emeryville, CA

David Smith - NOCCC
3030 Topaz No. A
Fullerton, CA 92631

NW PET USER'S GROUP
John F. Jones
2134 NE 45th. Avenue
Portland, OR 97213

Southeast Connecticut
Pet User Club
c/o Paul W. Sparks
13 Lincoln Dr.
Gales Ferry, Ct. 06335
203-464-6266

South Florida PUG
Dave Young
7170 SW 11th St.
W. Hollywood, FL 33023
(305) 987-6982

PET User Group
c/o MICH (Michigan
Computer Hackers)
2235 Lakeshore Drive
Muskegon MI 49441

St. Louis PET Users
(Mary Perkinson)
(314) 432-5225

Northern VA PET Users
2054 Eakins Ct.
Reston, VA 22091

Shelly Wernikoff
2731 N. Milwaukee Ave.
Chicago, IL 60647

SPHINX
(415) 584-3402

Vancouver PET User Group
Box 35353, Station E
Vancouver, BC, Canada

Lincoln Computer Club
750 E. Yosemite
Manteca, CA 95336

PET NET 1
14.24 MHz Sundays
10:00 A.M. Central time

PET NET II
7.205 MHz Fridays
9:00 A.M. Pacific time

PACS PET USER GROUP
20th & Olney Street
Phila. PA

PET LIBRARY
401 Monument Rd. No. 177
Jacksonville, FLA 32211

North London Hobby
Computer Club Press Release
The Polytechnic of North London
Department of Electronic and
Communications Engineering
Holloway, London N78DB

San Diego PUG
c/o D Costarakis
3562 Union St.
San Diego, CA 92103
(714) 235-7626 (7 am - 4 pm)

Independent PET Group
22 Firs Walk, Tewin Wood
Welsyn, Herts., UK

PET Users Group
2001 Bryan Tower Suite 3800
Dallas, TX 75201

Capital District PET Users
(Ben Green)
(518) 370-1820

PET Users of Japan
Soichiro Moridaira
Shinsen Park Himu, Rm. 150
4-13 Shensencho
Shibuyaku, Tokyo, Japan 150

David Liem
14361 Warwick Street
Detroit, MI 48223

Larry Williams
P.O. Box 652
San Antonio, TX 78293

Richard Prestien
6278 SW 14th Street
Miami, FL 33144

Long Island PET Society
Ralph Bressler
Harborfields HS
Taylor Ave.
Greenlawn, NY 11740

Utah Pug
Jack Fleck
2236 Washington Blvd.
Ogden, UT 84401

PET User Group
Doug Hennig
16 Everett Cres.
Regina, Sask. Canada
S4S 2M7

Gene Planchak
4820 Anne Lane
Sharpsville, PA 15150
(412) 962-9682

SEWPUG
Theodore J. Polczynski
P.O. Box 21851
Milwaukee, WI 53221
(414) 282-4181

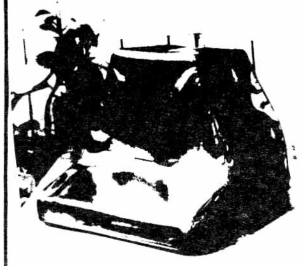
LAS VEGAS PUG
4743 Via San Rafael
Las Vegas, NV 89103
(John Melissa)

SEWPUG
(T J Polczynski)
3529 W Wanda Ave.
Milwaukee, WI 53221
282-4181

PUG of Westchester
(Bennett Meyer)
35 Barker Avenue
White Plains, NY 10601
914-428-7872

Triagle PUG
(Mike Adams)
218E Alexander Ave.
Durham, NC 27705
(919) 684-1891

Pawtucket PUG
(Scott Summer)
27 Leicester Way
Pawtecket, RI 02860



ITS COVERS . . .

A BEAUTIFUL COMPLEMENT TO ANY DECOR!

- Attractive, Durable, Leather-Grained Vinyl
- Double Stitching for Extra Strength
- Corded Seams Assure Perfect Fit
- Colors – Gold, Chestnut Brown, Olive Green or Black

PET/CBM – $18.95	*OTHER COMPUTER*
Apple II – $12.95	*COVERS AVAILABLE.*
Apple Disk – $8.95	*WRITE FOR DETAILS*

NEW-CURSOR FOR PET 2001 OLD ROM

INSTANT PUSHBUTTON CURSOR RETRIEVAL

$7.95

- Reviewed in Issue #1 of *COMPUTE*
- Restart without touching power
- Machine language programs in second cassette buffer not lost.
- Simple, illustrated instructions
- Installed in minutes – no soldering
- Improved mounting – no sticky tape

UNCRASHER for PET/CBM 2001 - New ROM — $14.9F

INTERNATIONAL TECHNICAL SYSTEMS, INC.
Box 264-a Woodbridge, Virginia 22194
Phone Orders (804) 262-9709
VISA master charge
SHIPPING AND HANDLING ADD $1.00

TRAP 65

For PET, APPLE, SYM, OSI, etc.
Any 1 mhz 6502!

- Prevents from "hanging up" via execution of **unimplemented opcodes!** Causes your monitor to **display location** of bad op code!
- Our computers have stopped crashing!
- **Debugging easier!**
- **Easy to install** — plugs into 6502's socket.

TRAP 65 is currently being used to insure that there are no bad opcodes in programs before submittal for ROM masks!

We are using TRAP 65 in experiments on extending the 6502's instruction set — PHX, PHY, INCD.

$149.95 (Add $4.00 postage, $10.00 foreign)

Eastern House Software

3239 Linda Dr. **Winston-Salem, N. C. 27106**

Finally, MAE — A PET DISK-Based MACRO ASSEMBLER/TEXT EDITOR

Works with 32K PET

- Works with 2040 Disk, and can drive 2022/2023 Printer, and/or RS232/20 ma Device thru User Port.
- 100% Disk Based, 100% Machine Language.
- Macros, Conditional Assembly, and a new feature we developed called Interactive Assembly.
- Coexists with Basic, Auto character repeat, Sorted Symbol Table.
- 27 Commands, 26 Pseudo Ops, 5 Conditional Ops, 38 Error Codes.
- Creates relocatable object code on disk.
- Assemble from Memory or Disk.
- String search, search and replace, and inter-line edit.
- Auto line #-ing, move, copy, delete, renumber.
- Labels up to 31 characters — user specifies length.
- Includes extention to PET monitor (disassemble, trace, etc.), Library of PET ROM locations, Relocating Loader, plus more.

Manual, Diskette, U. S. postage — $169.95
(Requires completion of License Agreement — Write for details)

Eastern House Software

3239 Linda Drive **Winston-Salem, N. C. 27106**

PET/CBM UNCRASHER™

WHAT IS IT? — UNCRASHER™ is a two button device that allows PET/CBM users to regain control of a cursor that's been lost due to programming errors. BASIC programs may be recovered. Machine language programs in the second cassette buffer are not disturbed either.

WHICH PETs/CBMs? — UNCRASHER™ is for all PET/CBM computers that use the "NEW" Version 2 ROMs. (Older model PETs should use the ITS NEW-CURSOR™.)

DOES IT WORK? — You bet!!! See the detailed review of the types of crashes and the concept of recovery in the first issue of *Compute*.

INSTALLATION — Simple, completely illustrated instructions using only a Phillips screwdriver ensure installation in minutes. No soldering or modificaitons to the computer.

OPERATION — Just follow the simple steps—push the buttons and reset the stack pointer—and PRESTO . . . recovery!

And all this happens without powering the PET/CBM down and up.

WHY UNCRASHER™? — No first class computer such as the PET/CBM should be without this capability. Whether your fancy be programming, business, education, or hobby, hobby, UNCRASHER™ saves you time by uncrashing your slip-ups.

AVAILABILITY — Now in better computer stores, or order direct from ITS, made by the people who brought you NEW-CURSOR™.

INTERNATIONAL TECHNICAL SYSTEMS INC.
P.O. BOX 264-a WOODBRIDGE, VIRGINIA 22194
CUSTOMER SERVICES –
RICHMOND, VIRGINIA (804) 262-9709
VISA master charge

PRODUCT MINI-REVIEWS

by Roy Busdiecker

COMMODORE PET 2001-32 Computer

After two years, I had gotten pretty used to my PET 2001-4 and all its idiosyncracies, so I took my time in deciding to buy a new system. Now that I've made the move, it seems to have been a good decision. Most impressive features in the new system are:

- correction of the bugs in the original PET interpreter (BASIC language)
- having enough user memory (RAM) to do reasonably large sized jobs (31744 bytes)
- "real" built-in keyboard
- green screen
- ability to do a "warm start" after losing control, without losing the BASIC program (this requires addition of a reset button like UNCRASHER, offered by ITS, Box 264, Woodbridge VA 22194. See Jim Butterfield's comments about it in the first issue of COMPUTE.).

Most distressing about the new machine was having to "fix up" programs that were built on the old PET. This applied not only to many of the programs I'd done myself, but also to some that are being offered by various commercial sources.

The body of the new machine is made of a type of plastic, but seems very sturdy. It's about a quarter-inch thick. It no longer has the handy prop rod used to "hold the hood up" on the old model when the body was opened...too bad.

Although it made a big dent in my bank account, it's worth it!

COMMODORE CBM 2040 DISK DRIVE

Using the PET Cassette, one of our programs was taking over five minutes to load. Each time we made changes, we'd save three copies, then verify them...and the process took more than half an hour!

With the CBM 2040, the same program takes less than 10 seconds to load! SAVEing takes about 14 seconds. The pain of working on long programs has virtually disappeared.

Since many of the early reviews on the 2040 reported serious reliability problems, I was reluctant to part with the sizeable investment required to purchase one; however, based on the assurance that the problems had been overcome, I took the leap. ...and am I gald I did!

From the moment the unit was unpacked and plugged in, it has worked properly...even though I thought it was fouled up for awhile (problem was solved, as a last resort, by reading the instructions!). Hint: any time you replace a disk, you must re-iritialize the location of the read-write heads.

working with the disk is a real pleasure, but it does require learning some new "magic words" and procedures. Commodore's manuals are getting better and better, and it wasn't too difficult to figure out what was required. It's not for the novice, but an intermediate or advanced programmer should be able to do what's needed.

The first program on the pre-programmed "demonstration disk" is a Disk Operating System (DOS) which provides a shorthand form of the special disk instructions. I recommend its use.

COMMODORE WORD PRO 2

WORD PRO 2 is Commodore's Word Processor package. Delivered in a plastic disk-storage box about 5" x 5" x 1½", the Word Pro 2 consists of one integrated circuit chip (ROM), one floppy disk, and an instruction booklet. In order to use it, you'll need a CBM 2040 disk drive, a PET or CBM 2001-16 or -32 (you can handle longer text segments with the larger memory computer), and a printer.

The instructions warn that you should have the ROM installed by your dealer...probably a good idea, to avoid any problems with your warranty. Since I have worked with integrated circuits, and knew what precautions to take, I decided to take the risk and do it myself...all went well, and there were no problems.

Once you get used to the package and its features, it's a very respectable word processor...not the best I've ever seen, but probably the best at such a reasonable price (it lists at $100). And it's far better than a typewriter! You can key in your text, review it on the screen, and make any necessary corrections, additions, or deletions before you have it printed.

There are provisions for tab stops, inserting or deleting words or lines, building letters out of standard parts, and moving groups of lines from one place to another in the letter (or article!). An automatic repeat key function is built in.

Two programs are provided...one for CBM printers with their unusual character set, and another for the ASCII character set used by most other printers.

Be prepared to spend a lot of time figuring out the instructions ...they're terrible! I had to prepare an instruction summary of my own to make the system usable.

FLASH! Commodore has announced a Word Pro III with significantly improved features over the WPII...and twice the price. It requires a 32K machine, also. A brief look at the new package reveals a product much closer to what you'd expect in a "real" word-processor. More later..............Roy Busdiecker

ATTENTION! Due to the terrific rise in the cost of paper, we must increase our subscription rates beginning March, 1980. Those of you who have already subscribed won't be affected this year by the increase, but new readers should be notified. Spread the word around, because in May we'll start returning $15.00 checks. According to our printer, we have enough paper on hand to handle April's issue....but the price of paper has increased nearly 75% over the last year. Sorry, friends.

REVIEW
NEW-CURSOR
$4.95
INTERNATIONAL TECHNICAL SYSTEMS, INC.
P.O. Box 264
Woodbridge, VA 22194

NEW-CURSOR is a momentary switch and resistor device which is designed to attach easily to your PET and give you the capability of a semi-warm reset. If you lose your cursor, a simple press on your NEW-CURSOR button will cause PET to reset without the shock to your power supply and video system such as you get when you turn your PET off and then on again.

The instructions provided are brief but clear. No soldering is required and the only tool needed is a screwdriver to open your PET. It took me (all thumbs) less than ten minutes to install my NEW-CURSOR which I received within a week of my order.

SURPRISE BONUS — I found that when I use NEW-CURSOR, I do not lose information stored in the 2nd cassette buffer!

This item is a MUST for anyone doing machine-language programming.

by Dr. Matarella

Review NEW-CURSOR
INTERNATIONAL TECHNICAL SYSTEMS,
Box 264
Woodbridge, VA 22194

Cursor, not to be confused with the cassette magazine of that name, is a reset button to clear a program or stop a crashed program without turning off the PET's power. This little $4.95 device consists of a pushbutton switch mounted with sticky tape and two jumpers with alligator clips — one grounded to a board mounting screw, the other going to a certain resistor on the board itself.

Installing cursor takes just a jiffy and it works exactly as advertised. One push of the button and you are back to the 'bytes free' message on the PET screen. Cursor is a worthwhile convenience and well-worth the price.

John Hirsch

Un-Crashing On Upgrade ROM Computers

Jim Butterfield, Toronto

If you do much work in machine language, sooner or later you'll write a program that will crash.

Formerly, you were out of luck. Unless you were lucky enough to stumble into a type 1 crash — which would take you to the Machine Language Monitor, or to an ?INVALID NUMERIC statement — your only remedy would be to reset, and wipe memory.

Type 2 crashes (tight loops) could be guarded against with a little preparation involving fiddling with the interrupt structure. But the nasty type 3 crash (X2 codes) cannot be fixed without kicking the Reset line; and Reset means memory test, and memory test means you'll have to reload your program.

No more. On upgrade ROMs, you can come out of a hard crash with memory preserved.

Method: Set the diagnostic sense pin to ground; then kick the Reset line. The processor will re-awaken in the Machine Language Monitor with memory preserved.

There's more: you're not yet out of the woods. Type a semicolon followed by RETURN; PET will respond with a question mark. Now move the cursor back to your register display line, and change the Stack Pointer (SP) value from 01 to F8. This strange procedure is important: you must follow it exactly. Once you've done so, you're clear. You may return to Basic with an X if you like, or proceed in the MLM.

Hardware: To make the diagnostic sense pin: take a standard 12-pin edge connector and wire pin 5 (diagnostic sense) to pin N (ground). Key the connector so it sits on the parallel user port. Plug it in whenever you want to un-crash, but don't leave it on the machine.

The Reset button is a little trickier, since you have to know where to connect it. Check with someone who's knowledgeable on PET hardware.

Commercial sources: International Technical Systems. Box 264, Woodbridge VA 22194 makes a Reset button.

THANK YOU FOR YOUR INTEREST IN THE

PROGRAMMER'S TOOLKIT RELOCATER PROGRAM

BUT . . . .

Nestar Systems, which owns Palo Alto ICs, who market the Programmer's Toolkit for the PET, have purchased all rights to the Programmer's Toolkit Relocater program that was announced in the *** PAPER. This product is therefore not available through me, and I am returning any payments you may have made as an enclosure with this notice.

Nestar Systems as owner of the Programmer's Toolkit Relocation Program may or may not choose to market this product. I believe that they will not be presently (1 Feb 80) marketing the Toolkit Relocater.

As part of my agreement with Nestar Systems, the following points are to be made to any Toolkit Relocater inquiries or orders:

1) I will not sell or make available the Toolkit Relocater program or equivalent products. I will not encourage the preparation of copies or relocated versions of the Programmer's Toolkit.

2) Those of you with addressing conflicts in the ROM are advised that several hardware manufacturers provide solutions for this problem. A few are:

"Socket 2 Me"	Skyles Electric Works
"Spacemaker"	Small Systems Service
"Dial-A-Rom"	Kobotek Systems

Advertisements by these manufacturers appear in the hobbyist/ personal computing journals, including Compute, Byte, Kilobaud, Creative Computing, Practical Computing and Printout.

3) If you make a copy of the Programmer's Toolkit to RAM, relocate the Programmer's Toolkit, or obtain in any manner copies or relocated copies of the Programmer's Tooolkit, you are in infringement of the copyright held by Nestar Systems and PAICS for the Programmer's Toolkit.

I personally feel that the Programmer's Toolkit is an excellent product and that we should all support Nestar by purchasing the official and genuine Toolkit, and by discouraging those who are making copies or distributing "bootleg" versions. As more ROM firmware will be available for the PET as time passes, the addressing conflicts which prompted your interest in the Toolkit Relocater will intensify. I suggest that you get one of the hardware products mentioned above in anticipation of this.

Thank you,

GREGORY YOB
(415) 326-4039

PO. Box 354
Palo Alto, CA.
94301

PET PROSE

There are many people who want specialized application programs to use on their PETs, but who are not knowledgeable enough to design or to write them.

If you can write significant software in some specialized field, and are willing to do so, we may be able to help you find the people who are willing to pay you to write programs for them. Send $25. with your name, address, and field of expertise. The information will be published in all the remaining issues of this volume.

Gene Polowytsch
72-31 67th Place
Glendale, NY 11385

QUALITY SOFTWARE FOR THE PET*

*PET is a product of Commodore Business Machines, Inc.

*** NOW AVAILABLE FOR MODELS 2001-8, 2001-16, and 2001-32 ***

PLEASE SPECIFY YOUR MODEL NUMBER WHEN ORDERING.

✦ SWEE-PING™

Test your eye-hand coordination as you control the "flying cursor" to create attractive patterns on the screen. Make the screen flash white (or black), make the cursor leave a trail of blocks or lines, or nothing at all ... white on black or reversed. Cursor bounces off "walls". Built-in scoring, review instructions on command. Develop creativity as you invent new games. If you like to program, this one is easy to modify and use as the basis for your own programs. A bargain at $5.95

✦ METRIC-CALC™

T PET into a powerful st calculator with exponent sc Met bet a ke ate. Look memories at one t ons than calculators costi mes as much. Unlike other conver, this one lets you use results in other calculations! $7.95

"... good METRIC ... conversions. The calculator is unique ... turn your PET into an everyday accessory. No problems in loading; the instructions are clear, and the program works. ... $7.95 is well-spent for METRICS alone."
John Kinast, PET Gazette, Spr 79

✦ CHEQUE-CHECK™

Take the worry a frustration out of balancing your c as you let this program guide step. "Remembers" entr find and correct mis ... even pro your checkboo enu- selection" o f time if you make o use, even beginners s a practical program that s every PET library. $7.95

"Check balancing is one of the few practical business applications suitable for the small computer. Of all the versions I have seen, this one is the best."
Robert Purser, CCM, May 79

✦ MEM-EXPLORER™

Want to see exactly how and where your programs are stored in PET's memory? It's a snap with this SOFTOOL KIT™ product which can be combined automatically with your own programs. Starting at the location you specify, it gives a screenful of information ... location address, contents in decimal, character or "token" interpretation, and double-byte address, integer forms. Based on the series of articles by Roy Busdiecker in The Paper, this version does even more! $7.95

✦ BILLBOARD™

Turn your store window or counter into getting advertising display! With BILLBOARD, you simply type in the choice (up to 254 characters), then see it march across the screen h letters ... even pause, or flash off and on if you like! A rea it costs less than a single ad. Order yours today! $49.95

"There are many billboard programs on the market today. This is the best one I have seen."
Robert Purser, CCM, May 79

BUY MSS PRODUCTS AT THESE FINE STORES:

The Computer Emporium, McLean, VA
Computers Plus, Alexandria, VA
Stonehenge Computer Co, Summit, NJ

Individual orders sent with payment directly to MSS are shipped via First Class Mail, at our expense, normally within 24 hours of receipt.

MICRO SOFTWARE SYSTEMS
P.O. Box 1442, Woodbridge, VA 22193

Send check or money order. Allow two weeks for check to clear. VA residents add 4% tax. Dealer inquiries invited.

INTERNATIONAL ORDERS: Pay by check drawn on US bank, or Int'l Money Order. Include shipping.

Riley Enterprises announces....

TOP Quality Vinyl Cassette Holders

12 Capacity in Black or Brown	$2.80
16 Capacity in Blk,Org or Lime	3.50

Add $2 for shipping orders under $20

Other types and colors are special order.

75 Pearson Street
Portsmouth, NH 03801 603-436-6564

Programs !!

● Business Research	$50
● Home & Small Business	$15-40
● Games & Simulations	$15 ea.
● Education (HS) Pack	$15

Send for complete catalog!

- Shopper
- Inventory
- Museum!
- Mansion!
- Pentagon!
- Home Addresser
- Deluxe Addresser
- Dinner's On!
- Fur Trapper
- High Seas

HARRY H. BRILEY
Catalog Info.
P.O. Box 2913
Livermore, CA.
94550
(415) 455 -9139

Please ship the following Software Shelf Programs to me within two weeks. I enclose payment in full **

PDPS FILE CREATE	$15.00 ____	WAREHOUSE	$ 7.95 ____
STAT III	$ 7.95 ____	CHASE	$10.00 ____
DATA RETRIEVAL	$10.00 ____	MICROMAZE	$ 7.95 ____
DATA EDIT	$10.00 ____	BLACKJACK	$10.00 ____
STAT I	$20.00 ____	BLOCKADE	$10.00 ____
STAT II	$20.00 ____	DEFLECTION	$10.00 ____
SPACE FIGHT	$10.00 ____	STARTREK 2001	$10.00 ____
HOSTAGE	$ 7.95 ____	XMON	$15.00 ____
AIR-SEA WAR	$ 7.95 ____	CMC/WPP	$29.50 ____
TUTOR	$19.95 ____	TUTOR PACKAGE	$39.95 ____
BASIC COMPLEAT	$29.95 ____	BAZAAR	$ 7.95 ____

I understand that if any of the programs fail to load, THE PAPER will send me a replacement as soon as I return the defective tape.

YES! Please enter my Subscription Order for all ten issues of

Vol I (1978) [ ]   Vol II (1979) [ ]   Vol III (1980) [ ]

NAME (Please print or type) ____________________

ADDRESS (Street, not P.O. Box) ____________________

CITY ____________________ STATE __________ ZIP __________

Please charge my /MC/VISA/BAC/ # ____________________

Master Charge Interbank # ____________________ Exp. Date __________

Required Credit Card Signature ____________________

THE PAPER
P.O. BOX 1142
COLUMBIA, MD 21044

SECOND CLASS POSTAGE
PAID IN COLUMBIA, MD 21044
(USPS 450-930)

TO: