POCKET COMPUTER NEWSLETTER

© Copyright 1982 — September Special Edition

THE SHARP PC-1500 AND RADIO SHACK PC-2 MACHINE LANGUAGE

The information contained in this document is the result of a truly monumental undertaking by **Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.** *It is the culmination of months of dedicated labor. In honor of his efforts,* **PCN** *proposes to refer to the mnemonics he has devised to describe the instruction set as the* **Rober Mnemonics.**

It must be realized by all who might utilize the results of the work described here, that all of the information was derived entirely through experimental procedures. No warrantee as to the accuracy or application of the information is made by the author or publisher. Persons not experienced in machine language procedures are probably best advised not to attempt learning it from the information contained herein. All prospective users are further warned that the incorrect application of machine language directives might damage peripheral devices connected to the PC. In other words, whatever use you care to make of the enclosed information is entirely at your own risk.

Here is a description of the CPU Registers and flags, the use of the BASIC "CALL" instruction, and a list of the mnemonics I have devised for the machine-language instructions used by the PC-1500 and PC-2.

A list of the codes for each instruction follows.

I wish to thank Nat Wadsworth and Brian Peterson for their suggestions, encouragement, and assistance in this rather lengthy project.

CPU Registers

A Accumulator (8 bits)

E Flag status Register (8 bits). Flac C (Carry) is bit 0; Flag I (Interrupt) is bit 1; Flag Z (Zero) is bit 2; Flag V (Overflow), bit 3; Flag H (Half-carry), bit 4.

X Index Register (16 bits). Contains a memory address; the data stored at that address will be designated as (X).

The high and low bytes of X are addressed separately as XH and XL.

Y Index Register, similar to X.

U Index Register, similar to X.

S Stack pointer (16 bits). Contains the RAM address of the next available position in the stack. Initialized at &784F by ROM instructions; it is decremented by 1 for each byte pushed onto the stack.

P Program counter (16 bits). Contains the address of the next machine-language instruction to be executed.

Uses of Flags

(1) Flags are used in conditional tests for branching and subroutine calls.

(2) The C flag is used in addition and subtraction as follows:

Addition: If Flag C is set, an extra 1 is added.

Subtraction: If Flag C is NOT set, an extra 1 is subtracted.

(3) In Rotate through Carry operations, the C flag is used.

Changes in Flag Status

The instructions affecting flags will clear or set them, as follows:

Flag C. In addition, the flag is set or cleared according to whether or not the resulting sum exceeds &FF. In subtraction, it is set if no borrow is required, and cleared if a borrow is required. The increments and decrements affecting this flag are treated as additions and subtractions. Bit rotation instructions also affect Flag C. Separate instructions exist to set or clear the flag.

Flag Z. When affected, Z is set when a result is zero, cleared otherwise.

Flag V. The operation of the V flag is based on twos–complement arithmetic in which a number from &00 to &7F is regarded as positive, and one from &80 to &FF as negative. In addition, V is set if the signs of the operands are alike *and* the sign of the resulting sum is opposite to that. In subtraction, it is set if the signs of the operands are opposite, *and* the sign of the resulting difference is opposite to that of the operand subtracted. Flag V is also affected by bit rotation instructions.

Flag H. This flag operates like a carry flag, except that it applies to carry and borrow between the two hex digits. Bit rotation also affects Flag H.

Calling a Machine Language Program

The instruction (in BASIC) CALL addr initiates execution of the machine language program at the memory address specified by the instruction.

The instruction (in BASIC) CALL addr, var has the same effect as the above, except for the following:

(1) The contents of the specified variable (a number from -32768 to 32767) will be placed into CPU register X.

(2) Whenever a machine-language subroutine return is executed while Flag C is set, the contents of CPU Register X will be transferred to the variable specified in the CALL statement, permitting the return of data directly to BASIC.

Two-Operand Instructions

Each of the following is completed by a specification of the data to be used as a second operand or the register or address containing that data.

STA	Store contents of A into the register or address specified.
SUBA	Data subtracted from A, result placed into A
ADDA	Data added to A, result placed into A
LDA	Load data into A.
CPA	Compare data to A. Flags are set the same as if the data had been subtracted from A, but A remains unchanged; flag C ignored in subtraction.
ANDA	Logical AND, result placed into A.
ORA	Logical OR, result placed into A.
EORA	Exclusive OR, result placed into A.
BITA	Bit test. Flags are set the same as if an AND had been performed, but A remains unchanged.
DSBA	Decimal subtraction from A (Binary-coded)
DADA	Decimal addition to A (Binary-coded)

In each of the above instructions, the second operand to be used is specified in one of the four ways illustrated by the examples below:

ADDA #nn	The hex digit pair nn, used as an operand, is part of the instruction.
ADDA XL	The operand is the contents of XL. (Note XH, YH, YL, UH, UL also used.)
ADDA (X)	The operand is the contents of the memory address pointed to by X; (Y) or (U) also used.
ADDAnnnn	The operand is the contents of memory address nnnn.

NOTE: There are other instructions similar to the above, in which A is replaced by some other CPU register or by a specified memory address.

Single-Operand Instructions.

INA Contents of A incremented by 1. (Note INXL, etc., also used.)
DEA Contents of A decremented by 1. (Note DEXL, etc. also used.)
PSHA Contents of A pushed onto stack.
POPA Contents of stack popped into A.

16-Bit Register Instructions

STX Y Contents of X stored into Y. (Note STZ, U, STX S, and STZ P also used.)
LDX Y X loaded with contents of Y. (Note LDX, U, LDX S, and LDX P also used.)
ADDX A Contents of A added to X., (Note ADDY A and ADDU A also used.)
INX X incremented by 1. (Note INY and INU also used.)
DEX X decremented by 1. (Note DEY and DEU also used.)
PSHX Contents of X pushed onto stack. (Note PSHY and PSHU also used.)
POPX Contents of stack popped into X., (Note POPY and POPU also used.)

Flag Instructions

CLRC Carry flag cleared.
SETC Carry flag set.

Two-Operation Instructions

STAI (X) A combination of STA (X) and INX, in that order.
STAD (X) A combination of STA (X(and DEX, in that order.
LDAI (X) A combination of LDA (X) and INX.
LDAD (X) A combination of LDA (X) and DEX.

Instructions also exist, similar to the above, using Index Registers Y and U.

CPAI (X) A combination of CPA (X) and INX.
INXY A combination of INX and INY.

Bit Rotation Instructions

In each, Flag Z will be set if the resulting number in A is zero. The effects on the accumulator and on flags C, V, and H are as follows:

RLA (Rotate Left.) V set if first bit of A changed.
RRA (Rotate Right). V set if last bit of A changed. .
RLCA (Rotate Left through Carry). V set if first bit of A changed.
RRCA (Rotate Right th ough Carry). V set if last bit of A changed.

Figure *Bit Rotation Instructions*

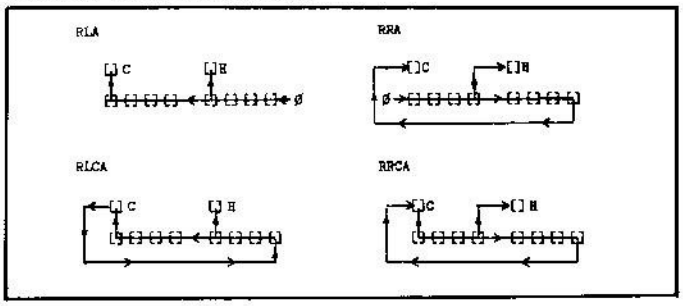
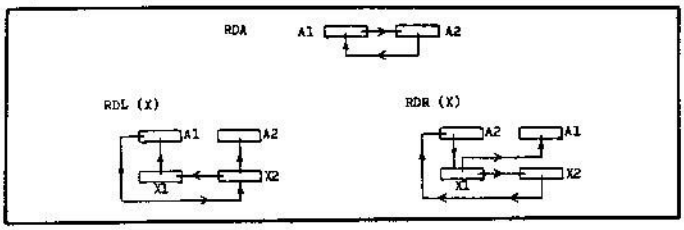


Figure *BCD Digit Rotation Instructions*



BCD Rotate Instructions

The first and second digits of A are denoted by A1 and A2; of (X), X1 and X2. Flags are not affected.

RDA (Rotate Digits of A).
RDL (X) (Rotate Digits Left).
RDR (X) (Rotate Digits Right).

Jumps and Branches

JMP nnnn A Jump to the address nnnn. (Program Counter is loaded with nnnn.)
FB #nn A relative Branch Forward by nn bytes, counting beginning at the address immediately following the FB #nn instruction. (Program counter is incremented by nn, in addition to the automatic increment following each instruction.)
RB #nn A relative Branch in Reverse, by nn bytes, counted as above.

The status of the flag tested determines whether a relative branch occurs in the following conditional branching instructions:

FBNC #nn Branch if C is not set. Note: Following (for example) CPA XL, this instruction may be interpreted as "Forward Branch if A is less than XL."
FBC #nn Branch if C is set. Following CPA XL, this instruction is equivalent to "Forward Branch is A is greater than or equal to XL."
FBNH #nn Branch if H is not set.
FBH #nn Branch if H is set.
FBNZ #nn Branch if Z is not set. Note: Following a CP, this may be interpreted as "Forward Branch if *not equal.*"
FBZ #nn Branch if Z is set. Following a CP, equivalent to "Forward Branch if *equal.*"
FBNV #nn Branch if V not set.
FBV #nn Branch if V is set.

The corresponding conditional Reverse Branches are also used.

BNZD #nn A special instruction to perform a relative Reverse Branch if UL is Not Zero and to decrement UL. The size of the relative branch is specified by nn.

Instructions for Subroutine Calls

JSR nnnn Jump to subroutine at memory address nnnn. (Return address is pushed onto stack, and program counter is loaded with nnnn.)
RTS Return from subroutine. (Return address is popped from stack into program counter.)

Base-Page Subroutine Calls

Page FF (addresses FF00 to FFFF in ROM) is used as a Base Page. It contains the beginning addresses of certain subroutines, which may be called as follows:

CALL #nn The 4-digit address of the subroutine called is listed on the base page, starting at FFnn.

Conditional calls of base-page subroutines, like conditional branches, use the status of a flag to determine whether the subroutine is called.

The following are included:

CANC #nn Call if C not set.
CAC #nn Call if C set.
CANH #nn Call if H not set.
CAG #nn Call if H set.
CANZ #nn Call if Z not set.
CAZ #nn Call if Z set.
CAV #nn Call if V set.

One-byte unconditional base page subroutine calls are also used.

Example: The op code F2, by itself, is equivalent to the instruction CALL #F2.

Note: Some of the base-page subroutines used in the ROM of the PC1500 pass paramenters to the subroutine, using the bytes that immediately follow the CALL instruction.

Additional Information:

SETI sets the interrupt flag

CLRI clears the interrupt flag

There are nine unnamed opcodes used in ROM, whose function I haven't yet determined. Information on these will be provided as soon as I have them figured out. (I hope.)

Here is what they are, with what I do know about them.

(1) Code 8A, in ROM at E22A only. Crashes in test program. (Could it be "Return from Interrupt"?)

(2,3) B8 and A8 codes are apparently related to each other; they occur in ROM in pairs, with B8 first, then (depending on a flag test, usually on Carry flag after a Bit Rotate) A8 may follow.

They are used in these locations in ROM:

B8: E1A0, E20A, E234, E2BE, E381, E4AA, ED71, EE63, FA69

A8: E1CB, E20D, E23B, E2C1, E384, E4C5, ED78, EE66, FA6C

(4) FD B1 code is used E2A8 only. In test program, stops everything dead; if paper feed key pressed, computer resumes, seemingly unaffected. (Could it be "Wait for Interrupt"?)

(5) FD BA is used at E427 only. (Possibly loading accumulator with code generated by pressing key on keyboard – untested guess.)

(6) FD 4C is used at CD81; and, beginning E359, used 3 times in succession!

In test program, switches computer off; when turned back on, "NEW0:CHECK" appears. It is a relatively "harmless" crash.

(7,8,9)	FD CE	Used	E004,	E230,	E2A4
	FD C1	Used	E00c,	E12A,	E313
	FD Co	Used	E006,	E012,	E30D

In a test program, this code FD CO will totally blank out the display (including BUSY, DEG, RUN, etc.) for about 5 seconds. Then the computer continues business as usual. I am baffled! While it's all blanked out, you can use the keyboard; what was typed in won't appear in the display until the computer has had 5 seconds following the last use of the keyboard.

Table Rober Mnemonics *and Hexadecimal Machine Codes for the PC-1500/PC-2.*

Instructions for Operations Using Accumulator

Instr	Operand Used with A										Flags Affected
	#nn	Address nnnn	Registers						Indexed			
			XH	XL	YH	YL	UH	UL	(X)	(Y)	(U)	
STA		AE nn nn	08	0A	18	1A	28	2A	0E	1E	2E	None
SUBA	B1 nn	A1 nn nn	80	00	90	10	A0	20	01	11	21	C,Z,V,H
ADDA	B3 nn	A3 nn nn	82	02	92	12	A2	22	03	13	23	C,Z,V,H
LDA	B5 nn	A5 nn nn	84	04	94	14	A4	24	05	15	25	Z only
CPA	B7 nn	A7 nn nn	86	06	96	16	A6	26	07	17	27	C,Z,V,H
ANDA	B9 nn	A9 nn nn							09	19	29	Z only
ORA	BB nn	AB nn nn							0B	1B	2B	Z only
EORA	BD nn	AD nn nn							0D	1D	2D	Z only
BITA	BF nn	AF nn nn							0F	1F	2F	Z only
DSBA									0C	1C	2C	C,Z,V,H
DADA									8C	9C	AC	C,Z,V,H

Instructions Using Other Registers

(Immediate data only--second operand is part of instruction.)

Instr	XH	XL	YH	YL	UH	UL	Flags
LD	48 nn	4A nn	58 nn	5A nn	68 nn	6A nn	None
CP	4C nn	4E nn	5C nn	5E nn	6C nn	6E nn	C,Z,V,H

Instructions Using Memory Addresses

(Immediate data only--second operand is part of instruction.)

Instr	Address nnnn	Indexed (X)	(Y)	(U)	Flags Affected
AND	E9 nnnn nn	49 nn	59 nn	69 nn	Z only
OR	EB nnnn nn	4B nn	5B nn	6B nn	Z only
BIT	ED nnnn nn	4D nn	5D nn	6D nn	Z only
ADD	EF nnnn nn	4F nn	5F nn	6F nn	C,Z,V,H

Instructions Addressing the Alternate Memory Buffer

When preceded by the code FD, any of the above instructions involving nnnn, (X), (Y), or (U) will apply to the alternate 64K of memory addressed by the CPU.

Two-Operation Instruction Codes

STAI (X)	41	STAI (Y)	51	STAI (U)	61	INXY	F5
STAD (X)	43	STAD (Y)	53	STAD (U)	63	CPAI (X)	F7
LDAI (X)	45	LDAI (Y)	55	LDAI (U)	65		
LDAD (X)	47	LDAD (Y)	57	LDAD (U)	67		

Jump Instruction Codes

JMP nnnn	BA nnnn
JSR nnnn	BE nnnn
RTN	9A

No-Operation Code

NOP	38

Increments and Decrements of Registers

Instr	Register							Flags Affected
	A	XH	XL	YH	YL	UH	UL	
IN	D0	FD 40	40	FD 50	50	FD 60	60	C,Z,V,H
DE	DF	FD 42	42	FD 52	52	FD 62	62	C,Z,V,H

Note: The previous status of the carry flag is ignored in the above.

Miscellaneous Instruction Codes

Instr	Op Code	Flags Aff
STA E	FD EC	none
LDA E	FD AA	Z only
PSEA	FD C8	none
POPA	FD 8A	Z only
SETC	FB	C only
CLRC	F9	C only

Instr	Op Code	Flags Aff
RLA	D9	C,Z,V,H
RRA	D5	C,Z,V,H
RLCA	DB	C,Z,V,H
RRCA	D1	C,Z,V,H
RDA	F1	none
RDL (X)	D7	none
RDR (X)	D3	none

Note: When preceded by code FD, the codes for RDL (X) and RDR (X) will apply to the alternate 64K memory buffer.

Sixteen-bit Instruction Op Codes

The flags are not affected or used by any of these instructions.

STX S	FD 4E	STX P	FD 5E	STX Y	FD 5A	STX U	FD 6A
LDX S	FD 48	LDX P	FD 58	LDX Y	FD 18	LDX U	FD 28

INX	44	INY	54	INU	64
DEX	46	DEY	56	DEU	66
ADDX A	FD CA	ADDY A	FD DA	ADDU A	FD EA
PSHX	FD 88	PSHY	FD 98	PSHU	FD A8
POPX	FD 0A	POPY	FD 1A	POPU	FD 2A
LDS #nnnn	AA nnnn				

Relative Branches and Base Page Calls

Forward Branches		Reverse Branches		Base-page Calls	
FB #nn	8E nn	RB #nn	9E nn	CALL #nn	CD nn
FBNC #nn	81 nn	RBNC #nn	91 nn	CANC #nn	C1 nn
FBC #nn	83 nn	RBC #nn	93 nn	CAC #nn	C3 nn
FBNH #nn	85 nn	RBNH #nn	95 nn	CANH #nn	C5 nn
FBH #nn	87 nn	RBH #nn	97 nn	CAH #nn	C7 nn
FBNZ #nn	89 nn	RBNZ #nn	99 nn	CANZ #nn	C9 nn
FBZ #nn	8B nn	RBZ #nn	9B nn	CAZ #nn	CB nn
FBNV #nn	8D nn	RBNV #nn	90 nn		
FBV #nn	8F nn	RBV #nn	9F nn	CAV #nn	CF nn

Special Conditional Reverse Branch:

BNZD #nn	88 nn

One-byte instructions for base-page subroutine calls are executed by

C0 C2 C4 C6 C8 CA CC CE D0 D2 D4 D6 D8 DA DC DE

E0 E2 E4 E6 E8 EA EC EE F0 F2 F4 F6 F8 FA FC FE

A DISASSEMBLER PROGRAM

This program decodes sequences of machine language instructions and converts them to assembly language using *Rober Mnemonics*. It may be used to print listings of hand-assembled machine language programs written by the user. Or, it may be used to list the instructions stored in ROM in the Sharp PC-1500 or Radio Shack PC-2.

You need at least a 4K RAM module in your PC in order to utilize this disassembler program.

Introductory Notes

To begin using the disassembler, execute RUN and respond to the initial prompts by entering the beginning and ending addresses of the code that is to be disassembled.

The disassembler uses the printer to produce a list of the assembly language instructions. As provided, output uses CSIZE 2 characters. It will, however, work with CSIZE 1 printing if desired.

Relative branches are calculated and the branch address is printed alongside the instruction.

As an option, a sequence of stored codes (machine codes) may be printed (using hexadecimal notation) by selecting DEF A. Enter the beginning and ending addresses when prompted.

Disassembling ROM

Before you can successfully disassemble ROM, you need to know where the instruction code sequences are located! Remember, not all of ROM contains instructions. Some of it holds various lookup tables. Attempting to disassemble these areas will simply yield nonsense. The following is a rudimentary map of ROM in the PC-1500, showing major instruction code and non-code areas:

C001 – C01C	Code
C01D – C3FF	Non-code (mostly tables)
C400 – D6AC	Code
D6AD – D6BE	Short lookup table
D5BF – DCAD	Code
DCAE – DCB5	Non-code in PC–1500, however, in the PC-2 addresses DCAE – DCB2 contain code.
DCB6 – E167	Code
E168 – E170	Short lookup table
E171 – F950	Code
F951 – F956	Non-code (???)
F957 – FBDF	Code
FBE0 – FFFF	Non-code (tables)

A similar map of ROM in the CE-150 printer/cassette interface is provided next:

A000 – A28A	Non-code (table used in printing)
A28B –AFF9	Code
AFFA – B009	Non-code
B00A – B015	? ? ? ?
B016 – B0EA	Non-code
B0EB – B7FF	Code
B800 – B809	Non-code
B80A – B81C	Code (some questionable areas)
B81D – B887	Non-code
B888 – BB55	Code
BB56 – BB69	Short lookup table
BB6A – BFFC	Code

The disassembler does not print the addresses of base-page calls. I did have it do so in an earlier version of the program, but it proved to be more of a hindrance than a help in attempting to follow the operation of a ROM routine. The accompanying table provides the addresses of the routines referred to by base-page calls. (The base page is FF.)

It must be noted that some routines in the ROM pass parameters to subroutines. Typically, one or more of the bytes immediately following a JSR or CALL instruction are used for that purpose. (When this occurs then the called subroutine modifies the return address to skip over these bytes.)

The disassembler program takes care of parameter-passing for base-page subroutine calls. However, some of the subroutines called by JSR instructions also pass parameters. When a JSR to a subroutine that may pass parameters is encountered, the PC will stop and display the prompt:

PASS?

The user must then enter the number of bytes to be passed. I am including a list of subroutines that are known to pass 1 byte. When the program requests "PASS?" information, check this list. If the subroutine (whose address will have just been printed) is in this list, enter the digit 1. If it is not, enter the digit 0 or just press the ENTER key.

List *Addresses of Subroutines Passing 1 Byte*

CC86	D2EC	DAB4
CC8B	D407	DB95
CC9C	D40D	DB83
D14C	D52A	DD2F
D14F	D6D9	DF9B
D2E0	D7CA	DFA0
D2EA	DAB2	DFA1

Tips on Interpreting Disassembled ROM

A subroutine may end with the CALL codes 48, 4A, 4C or 4E. If so, it is a subroutine that passes a parameter.

In many cases, a passed parameter is used to *increment* the return address, depending on the results of tests made within the subroutine. In the base-page calls having the base-page addresses: 00, 02, 04, 08, 0E, 1A, 28, 2C, 2E, C2, C4, C8, CE, D0, D2 and DE, the last byte passed may (or may not) be added to the return address.

The base-page subroutine called through base-page address 34 may select one of several passed values to modify the return address.

Additionally, each of the byte-passing subroutines *other than* the base-page ones, may or may not used the passed byte to increment the return address. I know of one exception: The subroutine at DD2F will *not* do this.

PC-1500/PC-2 Comparison

It is interesting to note that the ROM in the Radio Shack PC-2 is practically the same as that in the Sharp PC-1500. The few differences that do exist are probably minor revisions.

Table *Address of Subroutines Accessible through Base-Page FF.*

CALL:	ADDRESS:						
00	DCB7	40	C401	80	F707	C0	DD08
02	DCB6	42	CA58	82	F729	C2	DCD4
04	DCC6	44	CA7A	84	EF00	C4	DCD5
06	D065	46	CA80	86	EB40	C6	DD13
08	DDD9	48	DCF9	88	EDF6	C8	DCC5
0A	DE5E	4A	DCFD	8A	ED5B	CA	C001
0C	DE97	4C	DCE9	8C	EE1F	CC	DDC8
0E	D451	4E	DCED	8E	EDB1	CE	D45D
10	DD20	50	DA71	90	EDAB	D0	D5F9
12	DF93	52	F663	92	ED00	D2	DD1A
14	DFFA	54	F2B0	94	EC5C	D4	DEE3
16	DFF5	56	F23D	96	EA79	D6	DED1
18	DF80	58	F084	98	EC74	D8	DF3B
1A	D2E6	5A	E573	9A	ECEB	DA	C00E
1C	FA89	5C	F618	9C	ECB7	DC	DEBC
1E	FB2A	5E	F7A7	9E	E4A0	DE	D6DF
20	DF72	60	F684	A0	E234	E0	CD88
22	DF63	62	F88B	A2	E655	E2	C400
24	DEAF	64	F2B5	A4	B888	E4	CD89
26	DBB7	66	F2B9	A6	E451	E6	F70D
28	DBB1	68	F715	A8	B88B	E8	F661
2A	D03E	6A	F88F	AA	B88E	EA	F79C
2C	DCA6	6C	F6FB	AC	E88C	EC	F757
2E	D6C8	6E	F080	AE	B891	EE	F7CC
30	DC16	70	F747	B0	B894	F0	EF8A
32	D071	72	F7CE	B2	B897	F2	EE71
34	DF23	74	F775	B4	B89A	F4	DB8C
36	DF0F	76	F75F	B6	B89D	F6	DDB5
38	CE9F	78	F72F	B8	B8A0	F8	E171
3A	CFFB	7A	F70D	BA	F763	FA	E22C
3C	FA74	7C	F6E6	BC	E4B7	FC	E228
3E	FB8D	7E	F01A	BE	E4A8	FE	E000

```
10:INPUT "STARTIN
   G ADDRESS? ";A
11:INPUT "ENDING
   ADDRESS? ";B
12:CLS :ON ERROR
   GOTO 80
14:C=A:GOSUB 20:
   TAB 5:GOSUB 10
   0+PEEK A:A=A+1
   :LPRINT :IF A<
   =BGOTO 14
16:END
20:D=INT (C/256):
   GOSUB 24:D=C-2
   56*D:GOTO 24
22:A=A+1:D=PEEK A
24:E=INT (D/16):F
   =DAND 15:
   LPRINT CHR$ (E
   +48+7*(E>9));
   CHR$ (F+48+7*(
   F>9));:RETURN
26:GOSUB 22:GOTO
   22
30:TAB 10:LPRINT
   "#";:GOTO 22
32:TAB 15:LPRINT
   "#";:GOTO 22
34:TAB 10:GOTO 26
36:TAB 10:GOSUB 2
   6:GOTO 32
40:GOSUB 34:C=0:F
   =PEEK (A-1):IF
   F>208AND F<224
   OR F=204INPUT
   "PASS? ";C
42:CLS :IF C=0
   RETURN
44:LPRINT :TAB 5:
   LPRINT "PASS";
46:FOR I=1TO C:
   LPRINT " ";:
   GOSUB 22:NEXT
   I:RETURN
50:LPRINT "CALL "
   ;:D=PEEK A:
   GOSUB 24:GOTO
   54
52:GOSUB 30
54:C=0:IF D<47LET
   C=VAL MID$ ("3
   31010021000011
   000011211",D/2
   +1,1)
55:IF D=52LET C=3
   +2*PEEK (A+1)
56:IF D=194OR D=1
   96LET C=2+(
   PEEK (A+1)>223
   )
57:IF D>199LET C=
   VAL MID$ ("111
   22211000100000
   0020022000J",D
   /2-99,1)
58:GOTO 42
60:C=1:GOTO 64
62:C=-1
64:GOSUB 30:
   LPRINT ",";:C=
   A+1+C*D:GOTO 2
   0
70:LPRINT "ALT BU
   FFER:":TAB 5:
   GOTO 100+PEEK
   A
80:IF PEEK (A-1)=
   253LPRINT "FD
   ";
82:D=PEEK A:GOSUB
   24:LPRINT "??"
   :A=A+1:GOTO 14
90:"A"INPUT "STAR
   TING ADDRESS?
   ";A
91:INPUT "ENDING
   ADDRESS? ";B
92:CLS :LPRINT "B
   EGINNING ";:C=
   A:GOSUB 20:
   LPRINT ":":A=A
   -1:C=B-A:GOSUB
   46:LPRINT :END
100:LPRINT "SUBA X
   L";:RETURN
101:LPRINT "SUBA (
   X)";:RETURN
102:LPRINT "ADDA X
   L";:RETURN
103:LPRINT "ADDA (
   X)";:RETURN
104:LPRINT "LDA  X
   L";:RETURN
105:LPRINT "LDA  (
   X)";:RETURN
106:LPRINT "CPA  X
   L";:RETURN
107:LPRINT "CPA  (
   X)";:RETURN
108:LPRINT "STA  X
   H";:RETURN
109:LPRINT "ANDA (
   X)";:RETURN
110:LPRINT "STA  X
   L";:RETURN
111:LPRINT "ORA  (
   X)";:RETURN
112:LPRINT "DSBA (
   X)";:RETURN
113:LPRINT "EORA (
   X)";:RETURN
114:LPRINT "STA  (
   X)";:RETURN
115:LPRINT "BITA (
   X)";:RETURN
116:LPRINT "SUBA Y
   L";:RETURN
117:LPRINT "SUBA (
   Y)";:RETURN
118:LPRINT "ADDA Y
   L";:RETURN
119:LPRINT "ADDA (
   Y)";:RETURN
120:LPRINT "LDA  Y
   L";:RETURN
121:LPRINT "LDA  (
   Y)";:RETURN
122:LPRINT "CPA  Y
   L";:RETURN
123:LPRINT "CPA  (
   Y)";:RETURN
124:LPRINT "STA  Y
   H";:RETURN
125:LPRINT "ANDA (
   Y)";:RETURN
126:LPRINT "STA  Y
   L";:RETURN
127:LPRINT "ORA  (
   Y)";:RETURN
128:LPRINT "DSBA (
   Y)";:RETURN
129:LPRINT "EORA (
   Y)";:RETURN
130:LPRINT "STA  (
   Y)";:RETURN
131:LPRINT "BITA (
   Y)";:RETURN
132:LPRINT "SUBA U
   L";:RETURN
133:LPRINT "SUBA (
   U)";:RETURN
134:LPRINT "ADDA U
   L";:RETURN
135:LPRINT "ADDA (
   U)";:RETURN
136:LPRINT "LDA  U
   L";:RETURN
137:LPRINT "LDA  (
   U)";:RETURN
138:LPRINT "CPA  U
   L";:RETURN
```

```
139:LPRINT "CPA  (
    U)";:RETURN
140:LPRINT "STA  U
    H";:RETURN
141:LPRINT "ANDA (
    U)";:RETURN
142:LPRINT "STA  U
    L";:RETURN
143:LPRINT "ORA  (
    U)";:RETURN
144:LPRINT "DSBA (
    U)";:RETURN
145:LPRINT "EORA (
    U)";:RETURN
146:LPRINT "STA  (
    U)";:RETURN
147:LPRINT "BITA (
    U)";:RETURN
148:LPRINT "NOP";:
    RETURN
149:LPRINT "INXL";
    :RETURN
150:LPRINT "STAI (
    X)";:RETURN
151:LPRINT "DEXL";
    :RETURN
152:LPRINT "STAD (
    X)";:RETURN
153:LPRINT "INX";:
    RETURN
154:LPRINT "LDAI (
    X)";:RETURN
155:LPRINT "DEX";:
    RETURN
156:LPRINT "LDAD (
    X)";:RETURN
157:LPRINT "LDXH";
    :GOTO 30
158:LPRINT "AND  (
    X)";:GOTO 32
159:LPRINT "LDXL";
    :GOTO 30
160:LPRINT "OR   (
    X)";:GOTO 32
161:LPRINT "CPXH";
    :GOTO 30
162:LPRINT "BIT  (
    X)";:GOTO 32
163:LPRINT "CPXL";
    :GOTO 30
164:LPRINT "ADD  (
    X)";:GOTO 32
165:LPRINT "INYL";
    :RETURN
166:LPRINT "STAI (
    Y)";:RETURN
167:LPRINT "DEYL";
    :RETURN
168:LPRINT "STAD (
    Y)";:RETURN
169:LPRINT "INY";:
    RETURN
170:LPRINT "LDAI (
    Y)";:RETURN
171:LPRINT "DEY";:
    RETURN
172:LPRINT "LDAD (
    Y)";:RETURN
173:LPRINT "LDYH";
    :GOTO 30
174:LPRINT "AND  (
    Y)";:GOTO 32
175:LPRINT "LDYL";
    :GOTO 30
176:LPRINT "OR   (
    Y)";:GOTO 32
177:LPRINT "CPYH";
    :GOTO 30
178:LPRINT "BIT  (
    Y)";:GOTO 32
179:LPRINT "CPYL";
    :GOTO 30
180:LPRINT "ADD  (
    Y)";:GOTO 32
181:LPRINT "INUL";
    :RETURN
182:LPRINT "STAI (
    U)";:RETURN
183:LPRINT "DEUL";
    :RETURN
184:LPRINT "STAD (
    U)";:RETURN
185:LPRINT "INU";:
    RETURN
186:LPRINT "LDAI (
    U)";:RETURN
187:LPRINT "DEU";:
    RETURN
188:LPRINT "LDAD (
    U)";:RETURN
189:LPRINT "LDUH";
    :GOTO 30
190:LPRINT "AND  (
    U)";:GOTO 32
191:LPRINT "LDUL";
    :GOTO 30
192:LPRINT "OR   (
    U)";:GOTO 32
193:LPRINT "CPUH";
    :GOTO 30
194:LPRINT "BIT  (
    U)";:GOTO 32
195:LPRINT "CPUL";
    :GOTO 30
196:LPRINT "ADD  (
    U)";:GOTO 32
197:LPRINT "SUBA X
    H";:RETURN
198:LPRINT "FBNC";
    :GOTO 60
199:LPRINT "ADDA X
    H";:RETURN
200:LPRINT "FBC";:
    GOTO 60
201:LPRINT "LDA  X
    H";:RETURN
202:LPRINT "FBNH";
    :GOTO 60
203:LPRINT "CPA  X
    H";:RETURN
204:LPRINT "FBH";:
    GOTO 60
205:LPRINT "BNZD";
    :GOTO 62
206:LPRINT "FBNZ";
    :GOTO 60
207:LPRINT "RTI";:
    RETURN
208:LPRINT "FBZ";:
    GOTO 60
209:LPRINT "DADA (
    X)";:RETURN
210:LPRINT "FBNV";
    :GOTO 60
211:LPRINT "FB";:
    GOTO 60
228:LPRINT "FBV";:
    GOTO 60
229:LPRINT "SUBA Y
    H";:RETURN
230:LPRINT "RBNC";
    :GOTO 62
231:LPRINT "ADDA Y
    H";:RETURN
232:LPRINT "RBC";:
    GOTO 62
233:LPRINT "LDA  Y
    H";:RETURN
234:LPRINT "RBNH";
    :GOTO 62
235:LPRINT "CPA  Y
    H";:RETURN
236:LPRINT "RBH";:
    GOTO 62
237:LPRINT "RBNZ";
    :GOTO 62
238:LPRINT "RTS";:
    RETURN
239:LPRINT "RBZ";:
    GOTO 62
240:LPRINT "DADA (
    X)";:RETURN
241:LPRINT "RBNV";
    :GOTO 62
242:LPRINT "RB";:
    GOTO 62
243:LPRINT "RBV";:
    GOTO 62
244:LPRINT "SUBA Y
    H";:RETURN
245:LPRINT "RBNC";
    :GOTO 62
246:LPRINT "ADDA Y
    H";:RETURN
247:LPRINT "RBC";:
    GOTO 62
248:LPRINT "LDA  Y
    H";:RETURN
249:LPRINT "RBNH";
    :GOTO 62
250:LPRINT "CPA  Y
    H";:RETURN
251:LPRINT "RBH";:
    GOTO 62
253:LPRINT "RBNZ";
    :GOTO 62
254:LPRINT "RTS";:
    RETURN
255:LPRINT "RBZ";:
    GOTO 62
256:LPRINT "DADA (
```

```
    Y)";:RETURN
257:LPRINT "RBNV";
    :GOTO 62
258:LPRINT "RB";:
    GOTO 62
259:LPRINT "RBV";:
    GOTO 62
260:LPRINT "SUBA U
    H";:RETURN
261:LPRINT "SUBA";
    :GOTO 34
262:LPRINT "ADDA U
    H";:RETURN
263:LPRINT "ADDA";
    :GOTO 34
264:LPRINT "LDA  U
    H";:RETURN
265:LPRINT "LDA";:
    GOTO 34
266:LPRINT "CPA  U
    H";:RETURN
267:LPRINT "CPA";:
    GOTO 34
268:LPRINT "(A8)";
    :RETURN
269:LPRINT "ANDA";
    :GOTO 34
270:LPRINT "LDS  #
    ";:GOTO 26
271:LPRINT "ORA";:
    GOTO 34
272:LPRINT "DADA (
    U)";:RETURN
273:LPRINT "EORA";
    :GOTO 34
274:LPRINT "STA";:
    GOTO 34
275:LPRINT "BITA";
    :GOTO 34
277:LPRINT "SUBA";
    :GOTO 30
279:LPRINT "ADDA";
    :GOTO 30
281:LPRINT "LDA";:
    GOTO 30
283:LPRINT "CPA";:
    GOTO 30
284:LPRINT "(B8)";
    :RETURN
285:LPRINT "ANDA";
    :GOTO 30
286:LPRINT "JMP";:
    GOTO 34
287:LPRINT "ORA";:
    GOTO 30
289:LPRINT "EORA";
    :GOTO 30
290:LPRINT "JSR";:
    GOTO 40
291:LPRINT "BITA";
    :GOTO 30
292:GOTO 50
293:LPRINT "CANC";
    :GOTO 52
294:GOTO 50
295:LPRINT "CAC";:
    GOTO 52
296:GOTO 50
297:LPRINT "CANH";
    :GOTO 52
298:GOTO 50
299:LPRINT "CAH";:
    GOTO 52
300:GOTO 50
301:LPRINT "CANZ";
    :GOTO 52
302:GOTO 50
303:LPRINT "CAZ";:
    GOTO 52
304:GOTO 50
305:LPRINT "CALL";
    :GOTO 52
306:GOTO 50
307:LPRINT "CAV";:
    GOTO 52
308:GOTO 50
309:LPRINT "RRCA";
    :RETURN
310:GOTO 50
311:LPRINT "RDR  (
    X)";:RETURN
312:GOTO 50
313:LPRINT "RRA";:
    RETURN
314:GOTO 50
315:LPRINT "RDL  (
    X)";:RETURN
316:GOTO 50
317:LPRINT "RLA";:
    RETURN
318:GOTO 50
319:LPRINT "RLCA";
    :RETURN
320:GOTO 50
321:LPRINT "INA";:
    RETURN
322:GOTO 50
323:LPRINT "DEA";:
    RETURN
324:GOTO 50
326:GOTO 50
328:GOTO 50
330:GOTO 50
332:GOTO 50
333:LPRINT "AND";:
    GOTO 36
334:GOTO 50
335:LPRINT "OR";:
    GOTO 36
336:GOTO 50
337:LPRINT "BIT";:
    GOTO 36
338:GOTO 50
339:LPRINT "ADD";:
    GOTO 36
340:GOTO 50
341:LPRINT "RDA";:
    RETURN
342:GOTO 50
344:GOTO 50
345:LPRINT "INXY";
    :RETURN
346:GOTO 50
347:LPRINT "CPAI (
    X)";:RETURN
348:GOTO 50
349:LPRINT "CLRC";
    :RETURN
350:GOTO 50
351:LPRINT "SETC";
    :RETURN
352:GOTO 50
353:A=A+1.GOTO 400
    +PEEK A
354:GOTO 50
401:GOTO 20
403:GOTO 20
405:GOTO 20
407:GOTO 20
409:GOTO 20
410:LPRINT "POPX";
    :RETURN
411:GOTO 20
412:GOTO 20
413:GOTO 20
414:GOTO 20
415:GOTO 20
417:GOTO 20
419:GOTO 20
421:GOTO 20
423:GOTO 20
424:LPRINT "LDX  Y
    ";:RETURN
425:GOTO 20
426:LPRINT "POPY";
    :RETURN
427:GOTO 20
428:GOTO 20
429:GOTO 20
430:GOTO 20
431:GOTO 20
```

```
433:GOTO 70
435:GOTO 70
437:GOTO 70
439:GOTO 70
440:LPRINT "LDX  U
    ";:RETURN
441:GOTO 70
442:LPRINT "POPU";
    :RETURN
443:GOTO 70
444:GOTO 70
445:GOTO 70
446:GOTO 70
447:GOTO 70
464:LPRINT "INXH";
    :RETURN
466:LPRINT "DEXH";
    :RETURN
472:LPRINT "LDX  S
    ";:RETURN
473:GOTO 70
475:GOTO 70
476:LPRINT "(FD 4C
    )";:RETURN
477:GOTO 70
478:LPRINT "STX  S
    ";:RETURN
479:GOTO 70
488:LPRINT "INYH";
    :RETURN
492:LPRINT "DEYH";
    :RETURN
488:LPRINT "LDX  P
    ";:RETURN
489:GOTO 70
490:LPRINT "STX  Y
    ";:RETURN
491:GOTO 70
493:GOTO 70
494:LPRINT "STX  P
    ,";:RETURN
495:GOTO 70
496:LPRINT "INUH";
    :RETURN
498:LPRINT "DEUH";
    :RETURN
505:GOTO 70
506:LPRINT "STX  U
    ";:RETURN
507:GOTO 70
509:GOTO 70
511:GOTO 70
529:LPRINT "SETI";
    :RETURN
536:LPRINT "PSHX";
    :RETURN
538:LPRINT "POPA";
    :RETURN
540:GOTO 70
552:LPRINT "PSHY";
    :RETURN
556:GOTO 70
561:GOTO 70
563:GOTO 70
565:GOTO 70
567:GOTO 70
568:LPRINT "PSHU";
    :RETURN
569:GOTO 70
570:LPRINT "LDA  E
    ";:RETURN
571:GOTO 70
572:GOTO 70
573:GOTO 70
574:GOTO 70
575:GOTO 70
577:LPRINT "WAI";:
    RETURN
586:LPRINT "LDA  K
    B";:RETURN
590:LPRINT "CLRI";
    :RETURN
592:LPRINT "(FD C0
    )";:RETURN
593:LPRINT "(FD C1
    )";:RETURN
600:LPRINT "PSHA";
    :RETURN
602:LPRINT "ADDX A
    ";:RETURN
606:LPRINT "(FD CE
    )";:RETURN
611:GOTO 70
615:GOTO 70
618:LPRINT "ADDY A
    ";:RETURN
633:GOTO 70
634:LPRINT "ADDU A
    ";:RETURN
635:GOTO 70
636:LPRINT "STA  E
    ";:RETURN
637:GOTO 70
639:GOTO 70

STATUS 1
                5595
```


The POCKET COMPUTER NEWSLETTER is Available!

By Subscription Only: for a calendar year period (January – December). You get all issues published to date for the calendar year in which you subscribe, at the time you subscribe.

MC/VISA Phone Subscriptions: (203) 888-1946

☐ 1982 Regular Subscriber (Issues 11 – 20). $30.00 in U.S. (U.S. $36.00 to Canada. U.S. $45.00 elsewhere.)

☐ 1982/83 Subscriber (Issues 11 – 30). $60.00 in U.S. (U.S. $72.00 to Canada. U.S. $90.00 elsewhere.)

☐ 1983 Regular Subscriber (Issues 21 – 30). $36.00 in U.S. (U.S. $42.00 to Canada. U.S. $50.00 elsewhere.)

Orders must be accompanied by payment in full. We do not issue invoices for the POCKET COMPUTER NEWSLETTER.

Thank you for your remittance.

Name: _____

Addr: _____

City: _____ State: ____ Zip: _____

MC/VISA #: _____ Expires: _____

Signature: _____

P.O. Box 232, Seymour, CT 06483

LATEST FINDINGS

As this Special Edition of *PCN* went to press, Norlin reported that he had ascertained the coding for several more instructions. They are associated with the processing of interrupts and Input/Output operations as described here. The instructions with the mnemonics RTI, WAI and LDA KB have been incorporated into the disassembler listing provided on the preceeding pages.

Rober Mnemonic	Machine Code	Description
SETI	FD 81	Flag I set, interrupt enabled.
CLRI	FD BE	Flag I cleared, interrupt disabled.
RTI	8A	Return from interrupt.
WAI	FD B1	Wait for interrupt.
LDA KB	FD BA	Load accumulator with input from keyboard.

The final instruction in the list, LDA KB, results in a byte being placed in the accumulator. The byte loaded is determined by the row of the key matrix in which a key is depressed. The bit in the position corresponding to that row is 0, the other bits are 1.

What is Left?

Norlin reports that the following machine codes, which appear related to I/O operations, have not yet been fully defined. These codes do, however, appear in the PC-1500's ROM: A8, B8, FD 4C, FD C0, FD C1 and FD CE. Any ideas? Let Norlin know!