Printer Off
Radians
3 2 1 0
mode1
G2
Learn
Learn+Print
Step
List
Learn+Print
FP
3 2 1 0
mode0
K
K
CN-24
CN-36 (1 of 3)
XH
XL
XS(3)
K
A15
LEARN
PRIME
Keyboard
return
b3
call
b2
return
b2
call
b3
STK2
STK1
1
0
J
Prt
drum
pos.
A12
Prt
tach
N.C.
CN-24
Cass
data
A10
CN-36 (2 of 3)
Kbd
DH
DL
C4
C5
C
B9
0
0
1
2
3
4
5
6
7
G
H
L
0
1
2
3
4
5
6
7
F
0
1
2
3
4
5
6
7
E
Z
I
C
ALU
ACC
C
3
2
1
0
a
C0
B15
B
D
c
B12
B9
B14
Prog Err
Key Pressed
Mach Err
AH
AM
AL
C1
C2
C3
A*
A*
K
K
A2
A5
A3
A6
0
1
0
1
A11-8
4Kx4 RAM
A7-4
A3-0
ALE
RD
WR
MR
c
C6
b0
A4
A5
A6
A1
A2
A3
XR
c
Display
CN-36 (3 of 3)
4Kx4
Expansion
ROM(s)
0
1
2
3
4
5
6
7
C
C0
C1
C2
C3
C4
C5
C6
C7
PRIME
VERIFY PROG
SET P.C.
RECORD PROG
S.M.
B.S.
INS
DEL
000
001
002
003
004
005
006
007
NEXT
1
0
1
F7
return
b1
!return
c'
PC
2Kx42
ROM
008
K
RAM
size
F
E
NEXT
J
B
A
L
D
C
G
H
F
E
NEXT
J
B
K
A
D'
L
D
C
G
H
41 40 39
38 37 36
35 34 33 32 31 30 29 28 27
26
25 24 23 22
21 20 19 18
17 16 15 14
13
12
11 10 9
8 7 6
5 4 3
2 1 0

Clock timing:

a b b b b c

a = early latch, uCode load
b = serial data move
c = late latch, rd/wr RAM, latch uCode Addr

A op:
0: NOP?
1: wr(AH,AM,AL)
2: wr(15,15,AL)
3: wr(15,15, K)
4: rd(AH,AM,AL)
5: rd(15,15,AL)
6: rd(15,15, K)
7: Ld Prtr Hammer bits
8: Printer Feed
9: ?
10: DL = periph
11: Tape Data = bit 0 (DL)
12: DH,DL = Prt/Tape Sts
13: Tape Motor ON
14: Tape Motor OFF
15: XH,XL = <G>,<H>

B op:
0: NOP?
1: set bit 0 (ACC)
2: set bit 1 (ACC)
3: set bit 2 (ACC)
4: set bit 3 (ACC)
5: res bit 0 (ACC)
6: res bit 1 (ACC)
7: res bit 2 (ACC)
8: res bit 3 (ACC)
9: clear 6184
10: bit 0 (ACC) = !Z
11: bit 1 (ACC) = Z
12: Set Prog Err
13: clear ACC
14: Set Mach Err
15: ACC = (ALU) [if C0]

C op:
0: ACC = (ALU) [if B15]
1: AH = (ALU)
2: AM = (ALU)
3: AL = (ALU)
4: DH = (ALU)
5: DL = (ALU)
6: MR = (ALU)
7: NOP?

D op: (D'==0)
0: ALU = <G> + <H>
1: ALU = <G> + <H> + 1
2: ALU = <G> + <H>
3: ALU = <G> + <H> + C
4: ALU = <G> + <H> + 1
5: ALU = <G> & <H>
6: ALU = <G> | <H>
7: ALU = 0b0000

D op: (D'==1)
0: ALU = <H> - <G>
1: ALU = <H> - <G> - 1
2: ALU = <H> - <G>
3: ALU = <H> - <G> - C
4: ALU = <H> - <G> - 1
5: ALU = <G> & <H>
6: ALU = <G> ^ <H>
7: ALU = 0b0000

0-6 update Z flag
0-4 update I flag
2-4 update C flag

E op: conditional branch bit1
0: bit 0 (NEXT) = 0
1: bit 0 (NEXT) = 1
2: bit 0 (NEXT) = bit 0 (ACC)
3: bit 0 (NEXT) = bit 2 (ACC)
4: bit 0 (NEXT) = Prog Err
5: bit 0 (NEXT) = I
6: bit 0 (NEXT) = Any Key Down
7: bit 0 (NEXT) = Return

F op: conditional branch bit0
0: bit 1 (NEXT) = 0
1: bit 1 (NEXT) = 1
2: bit 1 (NEXT) = bit 1 (ACC)
3: bit 1 (NEXT) = bit 3 (ACC)
4: bit 1 (NEXT) = Z
5: bit 1 (NEXT) = ?
6: bit 1 (NEXT) = C
7: bit 1 (NEXT) = ?

G op:
0: 0b0000
1: K
2: mode 0*
3: mode 1
4: DH
5: DL
6: MR
7: XR

* Clears STEP

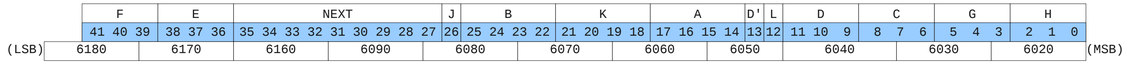
H op:
0: ACC
1: AH
2: AM
3: AL
4: DH
5: DL
6: MR
7: XR
L op 0: 0b000

J op: (F!=7)
0: jump
1: call

K op:
K = <imm>*

RAM size if PC=008

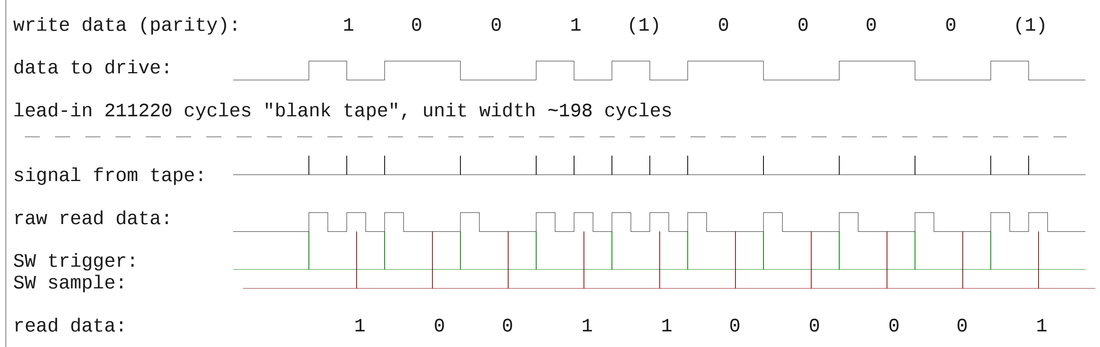
L op:
0: <H> = 0b0000
1: <H> = <H>



Keyboard direct jump
(microcode address)

PRIME	000
Verify Prog	001
Set PC	002
Rec Prog	003
Search Mk	004
B.S.	005
Insert	006
Delete	007

Tape Data
Example: 09 00 (Mark)



delta, trigger to sample: 272 cycles (ideal pulse width ~97 cycles)

RAM Addresses	Reg #	Prog Steps
0xfff - 0xff0	15 15	
0xfef - 0xfe0	15 14	
0xfdf - 0xfd0	15 13	
0xfcf - 0xfc0	15 12	
0xfbf - 0xfb0	15 11	
0xfaf - 0xfa0	15 10	
0xf9f - 0xf90	15 09	
0xf8f - 0xf80	15 08	
0xf7f - 0xf70	15 07	
0xf6f - 0xf60	15 06	0000-0007
0xf5f - 0xf50	15 05	0008-0015
0xf4f - 0xf40	15 04	0016-0023
0xf3f - 0xf30	15 03	0024-0031
...	...	...
0x13f - 0x130	01 03	1816-1823
0x12f - 0x120	01 02	1824-1831
0x11f - 0x110	01 01	1832-1839
0x10f - 0x100	01 00	1840-1847
0x0ff - 0x0f0	00 15	
0x0ef - 0x0e0	00 14	
0x0df - 0x0d0	00 13	
0x0cf - 0x0c0	00 12	
0x0bf - 0x0b0	00 11	
0x0af - 0x0a0	00 10	
0x09f - 0x090	00 09	
0x08f - 0x080	00 08	
0x07f - 0x070	00 07	
0x06f - 0x060	00 06	
0x05f - 0x050	00 05	
0x04f - 0x040	00 04	
0x03f - 0x030	00 03	
0x02f - 0x020	00 02	
0x01f - 0x010	00 01	
0x00f - 0x000	00 00	

System register usage

```
15 15  System memory
15 14  ?
15 13  "Scientific" display
15 12  "Floating Point" display
15 11  "Learn Mode" display
15 10  ?
15 09  (used for pi)
15 08  Prog Stack (4 words/entry)
15 07  Prog Stack
```

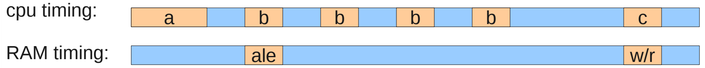
Prog Stack starts with 00 of 15 08 and proceeds to 15 of 15 08, then continues at 00 of 15 07.

Note, the system does not prevent use of these "registers", but using them almost certainly causes Mach Errs.

System memory "15 15"

```
15   bit0=Keytrace
14   (2..11 for prog step lo)
13   bit0=ProgTrace, bit1=GroupI/O
12   ?
11   Prog Stack Depth
10   ?
09   ?
08   Keyboard code, lo
07   Keyboard code, hi
06   ?
05   ?
04   ?
03   bit0=Run, bit1=GroupI/O
     bit2=ROM
02   Prog Step (RAM adr lo)
01   Prog Step (RAM adr mi)
00   Prog Step (RAM adr hi)
```

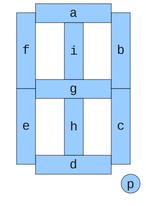
Clock timing:
a = early latch, uCode load
b = serial data move
c = late latch, rd/wr RAM, latch uCode Addr



NOTE: RAM Timing – AH,AM,AL are previous values (before cycle), MR is not overwritten until after cycle (read). I.e. Old values of AH,AM,AL are used, new value of MR is used (write).

Display segment decoder

```
     digits 1-12,14,15                digits 0,13
val  segments                  sym    segments                  sym
00   a b c d e f - - - -       '0'    - - - - - - g h i -       '+'
01   - - - - - - - h i -       '1'    - - - - - - g - - -       '-'
02   a b - d e - g - - -       '2'    - - - - - - g h i -       '+'
03   a b c d - - g - - -       '3'    - - - - - - g - - -       '-'
04   - b c - - f g - - -       '4'    - - - - - - g h i -       '+'
05   a - c d - f g - - -       '5'    - - - - - - g - - -       '-'
06   - - c d e f g - - -       '6'    - - - - - - g h i -       '+'
07   a b c - - - - - - -       '7'    - - - - - - g - - -       '-'
08   a b c d e f g - - -       '8'    - - - - - - g h i -       '+'
09   a b c - - f g - - -       '9'    - - - - - - g - - -       '-'
10   - - - - - - - - - p       '.'    - - - - - - g h i p       '+,'
11   - - c d - - g - - -       '>'    - - - - - - g - - -       '-'
12   - b - - - f g - - -       'u'    - - - - - - g h i -       '+'
13   a - - d - f g - - -       '≤'    - - - - - - g - - -       '-'
14   - - - d e f g - - -       '£'    - - - - - - g h i -       '+'
15   - - - - - - - - - -       ' '    - - - - - - - - - -       ' '
```



Digit positions map directly to AL of register "15 11"

Printer drum column-symbol decoder

val	columns 0-15	16	17	18	19	20
00	0	E	O	S	M	X
01	1	T	1	RE	ST	Y
02	2	+	2	W	α	Z
03	3	-	3	Go	SP	A
04	4	×	4	J○	Jø	B
05	5	÷	5	J+	J_{ε}	C
06	6	ST	6	SN	S^{-1}	D
07	7	RE	7	CS	C^{-1}	E
08	8	*	8	TN	T^{-1}	F
09	9	*	9	RD	DR	G
10	.	f	10	LN	LG	H
11	○	F	11	e^x	10^x	I
12	***	A	12	x^2	I	J
13	+	B	13	√x	|x|	K
14	-	C	14	LP	EP	L
15	Δ	D	15	$^1/_x$	RT	M

"***" spells "...OVERFLOW..." across the 16 columns

Commands by code

```
00 0d  "enter" digit d
00 10  Decimal point
00 11  Set Exp
00 12  Change Sign
00 13  ? Set mantissa sign '+'
00 14  Clear registers 00 00 - 00 15
00 15  Clear display (for entry)
01 rr  "Total" register rr
02 rr  Add register rr
03 rr  Subtract register rr
04 rr  Multiply register rr
05 rr  divide register rr
06 rr  Store register rr
07 rr  Recall register rr
08 ff  Program codes
09 ff  Program codes
10 xx  f(x) - call 10 xx
11 xx  F(x) - call 11 xx
12 xx  f(x) - call 10 xx in ROM
13 xx  F(x) - call 11 xx in ROM
14 rr  Exchange Display and
       register 00 rr.
15 xx  Special, extended, commands.
```

Program codes 08 xx

```
08 00  Search
08 01  Recall
08 02  Print
08 03  Go
08 04  Jump if zero
08 05  Jump if plus
08 06  sin(x)
08 07  cos(x)
08 08  tan(x)
08 09  rad->deg
08 10  natural log(x)
08 11  natural exp(x)
08 12  x squared
08 13  sqrt(x)
08 14  load prog
08 15  1/x
```

Program codes 09 xx

```
09 00  Mark
09 01  Store
09 02  alpha
09 03  Stop
09 04  Jump if not zero
09 05  Jump if error
09 06  inv sin(x)
09 07  inv cos(x)
09 08  inv tan(x)
09 09  deg->rad
09 10  base-10 log(x)
09 11  base-10 exp(x)
09 12  int(x)
09 13  abs(x)
09 14  end prog
09 15  return
```

"Program Print" (08 02)

```
When used in a program, "print"
requires an extra code to
specify the format:

-- 15  Paper feed (blank line)

tt pp  Print with letter tt,
       precision pp decimal
       places. (11-14 = sci.).

Letter Tags (tt):
00     X
01     Y
02     Z
03     A
04     B
05     C
06     D
07     E
08     F
09     G
10     H
11     I
12     J
13     K
14     L
15     M

See also 15 01.
```

"Alpha" (09 02) codes

```
08 02  (alpha print)
       trace keystrokes
09 02  (alpha alpha) Trace off
08 00  (alpha search) pi
09 03  (alpha stop) pause 0.5 sec
```

15 xx Commands by code

```
15 00 rr rr   Recall?
15 01 pp pp   Program Print (works
              from keyboard)
15 02 ii ii   I/O
15 03 mm mm   ROM command
15 04 mm mm   ROM command
15 05 mm mm   ROM command
15 06 mm mm   ROM command
15 07 mm mm   Call mark?
15 08 nn nn   ?
15 09 nn nn   ?
15 10 aa aa   Alpha?
15 11 cc cc   Indir
15 12 mm mm   ROM command
15 13 ii ii   Group 1 (XS=4)
15 14 ii ii   Group 2 (XS=5)
15 15 nn nn   ?
```

15 02 (I/O) Command suffix

```
00 xx  CN-24 Print (XS=1)
01 xx  CN-24 Print (XS=1)
02 xx  CN-24 Print (XS=1)
03 xx  CN-24 Print (XS=1)
04 xx  CN-24 Print (XS=1)
05 xx  CN-24 Print (XS=1)
06 xx  CN-24 Print (XS=1)
07 xx  CN-24 Print (XS=1)
08 xx  CN-24 Print (XS=1)
09 xx  CN-24 Print (XS=1)
10 xx  ?
11 xx  ?
12 xx  CN-24 Print (XS=1)
13 xx  CN-36 I/O? (XS=2/3)
14 xx  ?
15 xx  ?
```

CN-24: Print (IBM Selectric?) Command suffix

```
12 00 Print CR-LF?
12 xx Print xx blanks

yy = 01..09
xx = 00..14
yy xx  Print display, formatted
       yy whole-number width,
       xx decimal places.

yy 15  Print display, scientific
       notation.

00 --  Print display, scientific
       notation.

Character Codes:

00 00  '-' (minus)
00 02  ' ' (blank)
01 06  '.' (decimal point)
02 05  '+' (plus)
02 08  (unknown)
02 09  '1'
03 00  '9'
03 01  '0'
03 04  '6'
03 05  '5'
03 06  '2'
03 09  '4'
03 12  '8'
03 13  '7'
03 14  '3'
```

ROM Targeting Commands

```
12 xx  f(x) in ROM
13 xx  F(x) in ROM
15 03  SEARCH in ROM (?)
15 04  SEARCH in ROM (?)
15 05  SEARCH in ROM (?)
15 06  SEARCH in ROM (?)
15 12  SEARCH in ROM (?)
```