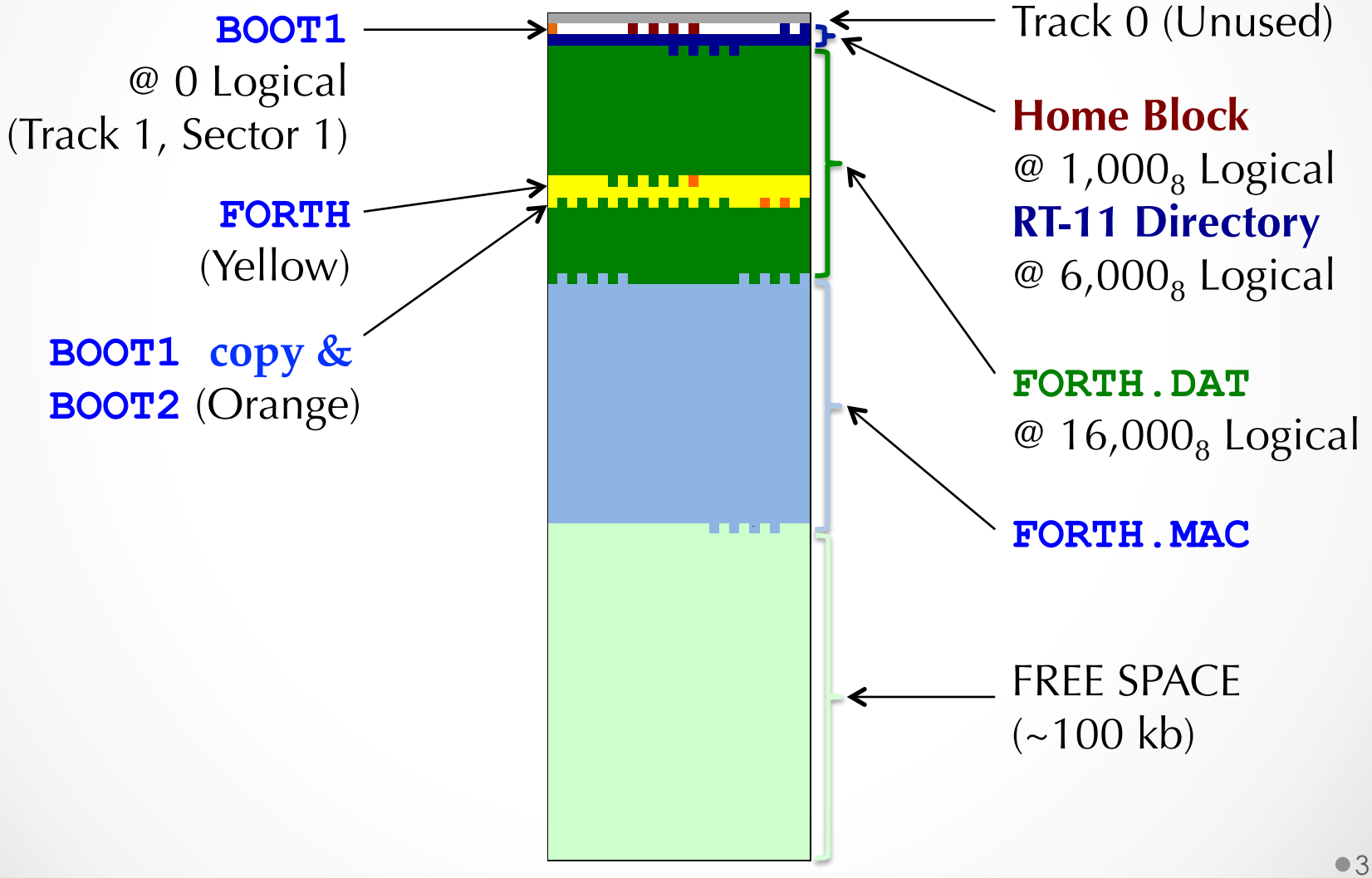
Reconstructing the fig-FORTH PDP-11 Disk

Paul Hardy, 25 March 2017

Good News & Bad News

- The Good News: FORTH.DAT contains screens to create a disk with a standalone Forth executable
- The Bad News: they don't work unless you're already running standalone on 2 RX01 floppy disks

FIG RX01 Disk Layout



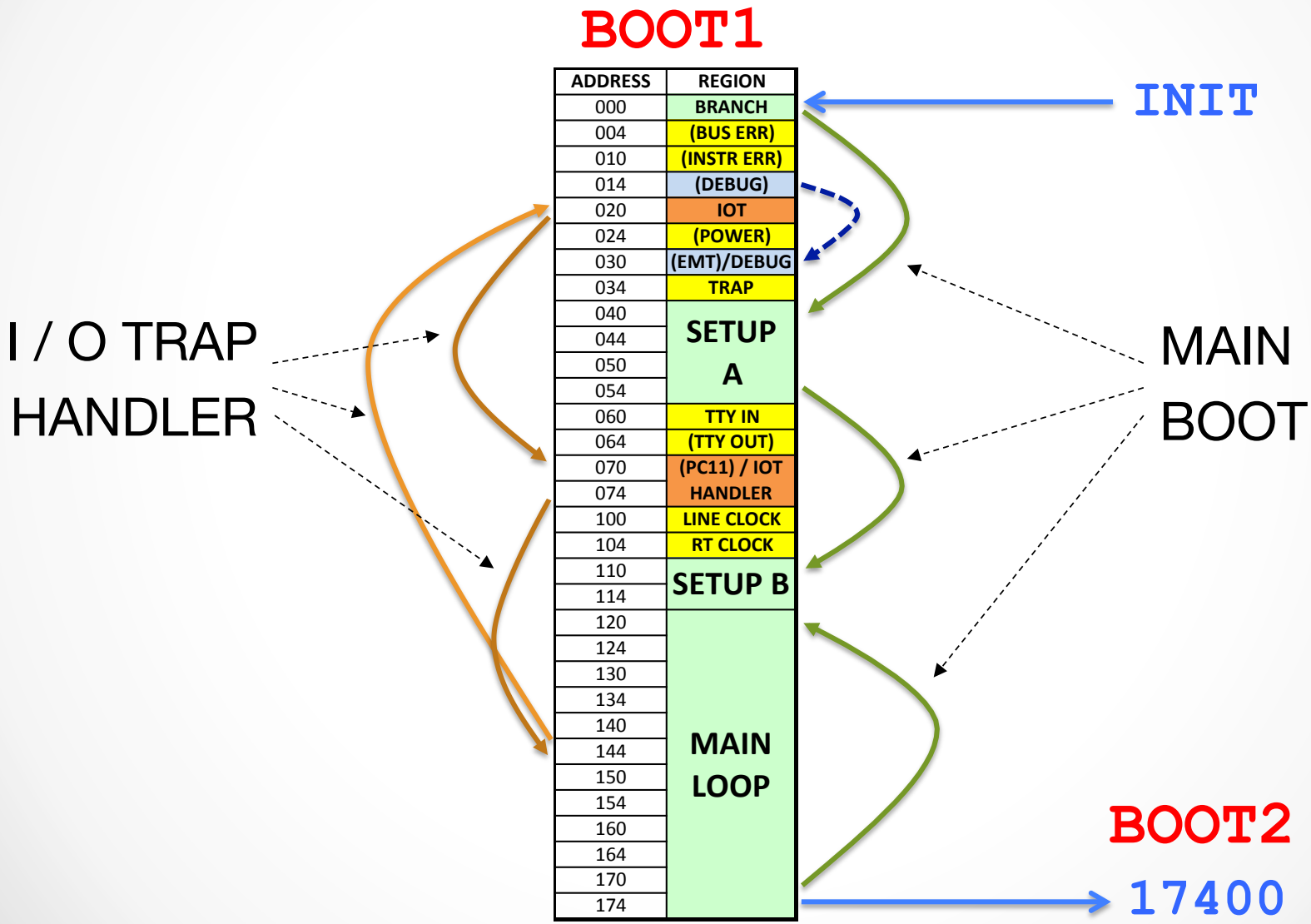
You Can't Get There from Here

- FORTH.DAT-created binary image:
  - Command: **`34 LOAD`**
  - Do not load other screens first — minimize dictionary size
  - Ignore "`MSG # 4`" (word not unique) warnings
  - **ISSUES:**
    - Copies memory image with OS info to FORTH.DAT
    - 2-sector boot loader written to RX01 #2, not FORTH.DAT
- Where <u>can</u> we get to from here?

It is a riddle,
wrapped in a mystery,
inside an enigma;
but perhaps there is a key.

—Winston Churchill

GOTO Considered *Sine Qua Non*



Screen 36: Boot Loader

```
SCR # 36
  0 ( CREATE BOOT LOADER.  NOTE - DOES NOT WRITE BOOT BLOCK)
  1 ASSEMBLER DEFINITIONS OCTAL
  2 : INIT, 1000 # R0 MOV,   00000 # R1 MOV,
  3    177170 # R4 MOV,   200 # R3 MOV, ;
  4 : ?TERM, R1 () TSTB,   LE IF, 1000 @# JMP, ENDIF, ;
  5 : WAITT, BEGIN, R3 R4 () BIT, NE UNTIL, ;
  6 : WAITD, BEGIN, 40 # R4 () BIT, NE UNTIL, ;
  7 : ?ERR, R4 () TST, LE IF, HALT, ENDIF, ;
  8 : BLOOP, R3 R2 MOV,
  9    BEGIN, WAITT, 2 R4 I) R0 )+ MOVB,   R2 DEC,   EQ UNTIL, ;
 10 : NEXTTAB, 1 R1 I) R5 MOVB,   R5 INC, R5 INC,
 11    R5 32 # CMP, GT IF, 32 # R5 SUB, THEN,
 12    R5 1 R1 I) MOVB,   1 R1 I) 2 R1 I) CMPB,
 13    EQ IF, 3 # R1 ADD, ENDIF,  ;
 14
 15    DECIMAL 37 LOAD
 OK
```

The Enigmatic TABLE,

```
SCR # 37
  0 ( CREATE BOOT LOADER, CONT.)       OCTAL
  1 : TRACK, R1 () R5 MOVB, R5 2 R4 I) MOV, ;
  2 : SECTOR, 1 R1 I) R5 MOVB,   R5 2 R4 I) MOV, ;
  3 : MAINL, BEGIN, ?TERM, 7 # R4 () MOV, WAITT, SECTOR, WAITT,
  4    TRACK, WAITD, ?ERR,  3 # R4 () MOV,  BLOOP, NEXTTAB,
  5    400 UNTIL, ;
  6 : 2, 400 * + , ;
  7 : TABLE,   17 27 2,   7 17 2,   10 10 2,
  8    20 15 2,   15 20 2,   16 16 2,
  9    21 23 2,   23 21 2,   24 26 2,   0 0 2,  ;
 10
 11 CODE BOOT 35000 JMP, C;
 12 : TASK ;
 13 35000 DP ! HERE 6 + INIT, WAITD, MAINL, HERE SWAP ! TABLE,
 14 FORGET TASK
 15 17572 35006 !   35000 21 26 WTS  35200 21 30 WTS
 OK
```

The Mysterious 17572

```
1 LOAD

LOADING EDITOR... R ISN'T UNIQUE I ISN'T UNIQUE
LOADING ASSEMBLER... R0 ISN'T UNIQUE # ISN'T UNIQUE
LOADING STRING PACKAGE...
BYE ISN'T UNIQUE
 OK
36 LOAD
TASK ISN'T UNIQUE WTS ?
35000 @ U. 35002 @ U. 35004 @ U. 35006 @ U.
12700 1000 12701 17572  OK
35010 @ U. 35012 @ U. 35014 @ U. 35016 @ U.
12704 177170 12703 200  OK
DECIMAL 36 LIST

SCR # 36
  0 ( CREATE BOOT LOADER.  NOTE - DOES NOT WRITE BOOT BLOCK)
  1 ASSEMBLER DEFINITIONS OCTAL
  2 : INIT, 1000 # R0 MOV,   00000 # R1 MOV,
  3    177170 # R4 MOV,   200 # R3 MOV, ;
```

Boot Loader Dump

```
: MEM. OCTAL DO I 2 * 35000 + DUP U. @ U. CR LOOP ;
 OK
10 0 MEM.
35000 12700
35002 1000
35004 12701
35006 17572
35010 12704
35012 177170
35014 12703
35016 200
 OK
```

```
35000 @ U. 35002 @ U. 35004 @ U. 35006 @ U.
12700 1000 12701 17572  OK
35010 @ U. 35012 @ U. 35014 @ U. 35016 @ U.
12704 177170 12703 200  OK
```

Eureka!

```
SCR # 37
  0 ( CREATE BOOT LOADER, CONT.)       OCTAL
  1 : TRACK, R1 () R5 MOVB, R5 2 R4 I) MOV, ;
  2 : SECTOR, 1 R1 I) R5 MOVB,   R5 2 R4 I) MOV, ;
  3 : MAINL, BEGIN, ?TERM, 7 # R4 () MOV, WAITT, SECTOR, WAITT,
  4    TRACK, WAITD, ?ERR,  3 # R4 () MOV,  BLOOP, NEXTTAB,
  5    400 UNTIL, ;
  6 : 2, 400 * + , ;
  7 : TABLE,   17 27 2,   7 17 2,   10 10 2,
  8    20 15 2,   15 20 2,   16 16 2,
  9    21 23 2,   23 21 2,   24 26 2,   0 0 2,  ;
 10
 11 CODE BOOT 35000 JMP, C;
 12 : TASK ;
 13 35000 DP ! HERE 6 + INIT, WAITD, MAINL, HERE SWAP ! TABLE,
 14 FORGET TASK
 15 17572 35006 !   35000 21 26 WTS  35200 21 30 WTS
 OK
```

TABLE, Decoded (Octal)

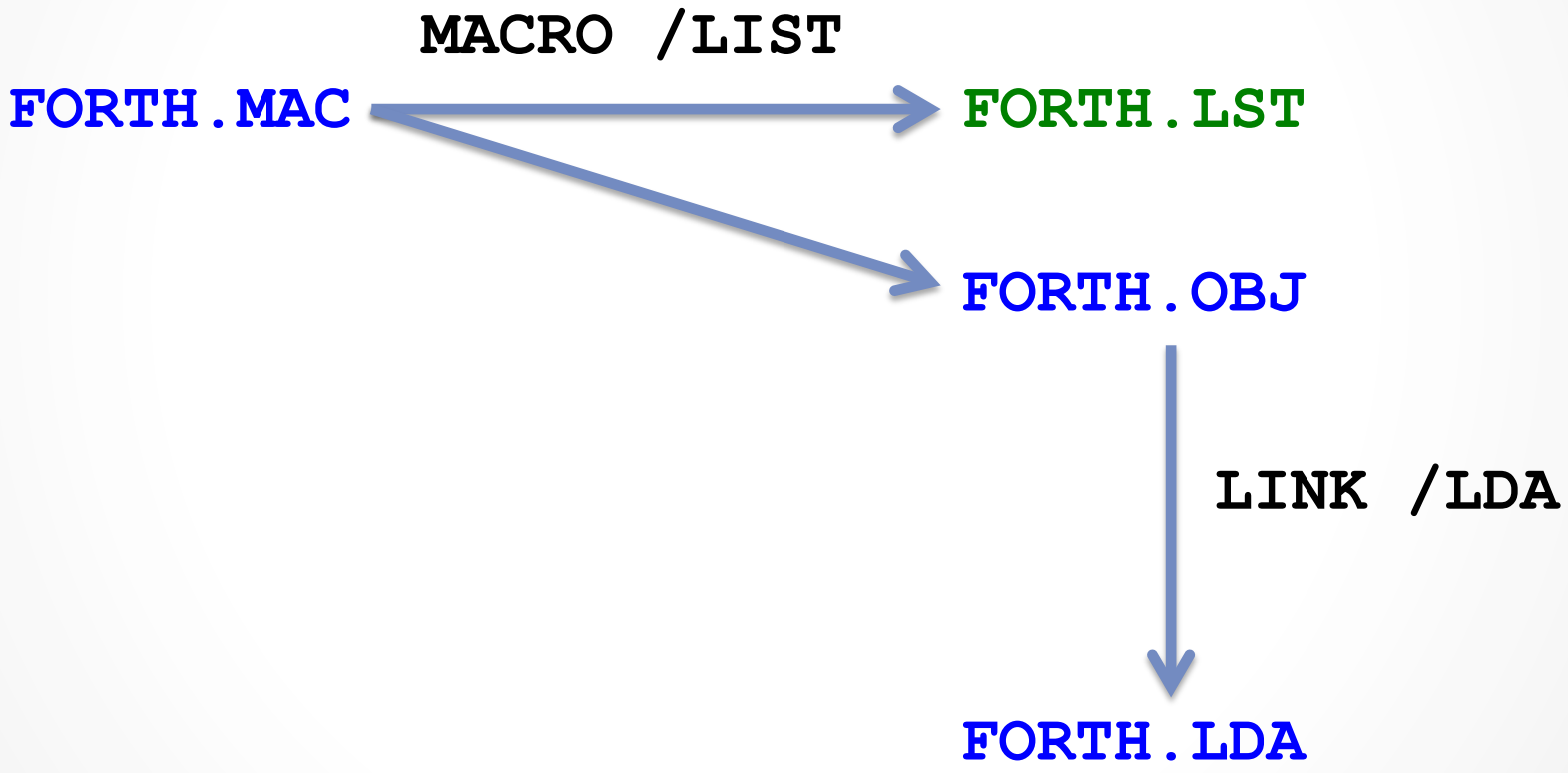
FORTH.DAT Screens 40 through 47													
Track	**Physical Sectors**												
17					17	21	23	25	27	31	1	3	5
17	10	12	14	16	20	22	24	26	30	32	2	4	6
20	15	17	21	23	25	27	31	1	3	5	7	11	13
20	16	20	22	24	26	30	32	2	4	6	10	12	14
21	23	25	27	31	1	3	5	7	11	13	15	17	21
21	24	26	30										

```
: TABLE,
  17     27 2,  7
  17 2,  10     10 2,
  20     15 2,  15
  20 2,  16     16 2,
  21     23 2,  23
  21 2,  24     26 2,
  0      0  2,  ;
```

Binary Images

- BOOT0: Optional Console Bootstrap
- BOOT1: Init Boot Sector (Track 1, Sector 1)
- BOOT2: 2-Sector Boot Loader
- FORTHS: Standalone Forth Binary

Compiling on RT-11



LDA Frame Format

Byte	Content
1	1
2	0
3	Byte Count, Low Byte
4	Byte Count, High Byte
5	Load Address, Low Byte
6	Load Address, High Byte
	Data Bytes
	Checksum Byte

Last Frame (6 Bytes) Contains Even Start Address

All Frame Bytes + Checksum = 0 (2's Complement)

make prep

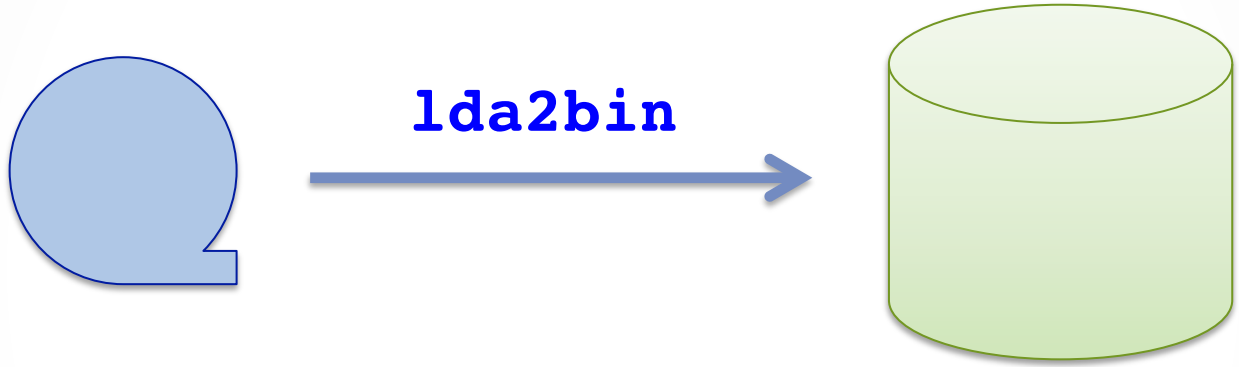
```
PDP_SRC=RX.COM BOOT0.MAC BOOT1.MAC \
        BOOT2.MAC FORTHS.MAC

prep: crlf $(PDP_SRC)
        for i in $(PDP_SRC) ; do \
            ./crlf < $$i > $$i.crlf ; \
            mv $$i.crlf $$i ; \
        done
        cp FORTHS.MAC FORTH.MAC
        touch prep
```

RX.COM

```
MACRO  BOOT0   /LIST
LINK   BOOT0   /LDA /TRANSFER:1000
MACRO  BOOT1   /LIST
LINK   BOOT1   /LDA /TRANSFER:0
MACRO  BOOT2   /LIST
LINK   BOOT2   /LDA /TRANSFER:17400
MACRO  FORTHS  /LIST
LINK   FORTHS  /LDA /TRANSFER:1000
```

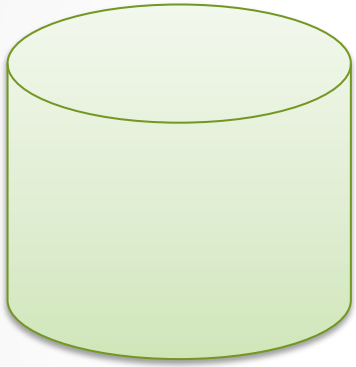
Type "**@RX**" to invoke under RT-11

Extracting Binary Images



BOOT0.LDA	boot0.bin
BOOT1.LDA	boot1.bin
BOOT2.LDA	boot2.bin
FORTHS.LDA	forths.bin

Standalone fig-FORTH



makerx

FORTH.DAT
FORTH.MAC
} RT-11 Directory

boot1.bin

boot2.bin

forths.bin

make rx01

```
./lda2bin < BOOT0.LDA > boot0.bin
   Memory range is [001000, 001043]
   Execution begins at 001000
./lda2bin < BOOT1.LDA > boot1.bin
   Memory range is [000000, 000175]
   Execution begins at 000000
./lda2bin < BOOT2.LDA > boot2.bin
   Memory range is [017400, 017614]
   Execution begins at 017400
./lda2bin < FORTHS.LDA > forths.bin
   Memory range is [001000, 044153]
   Execution begins at 001000
./makerx > rx01_figforth-1.3.1.dsk
```

lda2bin -v

```
bash-3.2$ ./lda2bin -v < FORTHS.LDA > /dev/null^M
   Start: 001000, Bytes: 46 octal.^M
   Start: 001046, Bytes: 10 octal.^M
   Start: 001055, Bytes: 23 octal.^M
   Start: 001077, Bytes: 43 octal.^M
   Start: 001141, Bytes: 45 octal.^M
   Start: 001206, Bytes: 40 octal.^M
   Start: 001245, Bytes: 45 octal.^M
```

-
-
-

lda2bin -v (cont'd)

```
   Start: 016403, Bytes: 31 octal.^M
   Start: 036140, Bytes: 02 octal.^M
    Mem @ 036140 = 000^M
    Mem @ 036141 = 000^M
   Start: 040142, Bytes: 04 octal.^M
    Mem @ 040142 = 000^M
    Mem @ 040143 = 000^M
    Mem @ 040144 = 000^M
    Mem @ 040145 = 000^M
   Start: 042146, Bytes: 04 octal.^M
    Mem @ 042146 = 000^M
    Mem @ 042147 = 000^M
    Mem @ 042150 = 000^M
    Mem @ 042151 = 000^M
   Start: 044152, Bytes: 02 octal.^M
    Mem @ 044152 = 000^M
    Mem @ 044153 = 000^M
   Start: 001000, Bytes: 00 octal.^M
   Memory range is [001000, 044153]^M
   Execution begins at 001000^M
bash-3.2$ exit^M
```

FORTHS.LST: **DSKBUF:**

```
26 035140  000000   .WORD   0
27 035142           .BLKB   1024.
28 037142  000000   .WORD   0
29 037144  000000   .WORD   0
30 037146           .BLKB   1024.
31 041146  000000   .WORD   0
32 041150  000000   .WORD   0
33 041152           .BLKB   1024.
34 043152  000000   .WORD   0
35 043154
```

.WORD → .BLKW

```
;
DSKBUF:         ; ROOM  FOR 3 1K DISK BUFFERS
; INITIALIZE BUFFERS' UPDATE BITS, AND TERMINATING NULLS, TO ZERO.
; NOTE - THESE BUFFERS ARE CLEARED AT COLD START, ANYWAY,
; BECAUSE A STAND-ALONE BOOT MAY NOT INITIALIZE HIGH MEMORY;
; AND ALSO SO THAT THE NUMBER OR LOCATION OF BUFFERS CAN BE
; CHANGED AT RUN TIME.
        .BLKW   1
        .BLKB   1024.
        .BLKW   1
        .BLKW   1
        .BLKB   1024.
        .BLKW   1
        .BLKW   1
        .BLKB   1024.
        .BLKW   1
ENDBUF:         ; CAUTION - 'ENDBUF' - 'DSKBUF' MUST BE EXACT MULTIPLE
; OF THE BUFFER LENGTH PLUS 4.
;
```

make rx01 (2nd Try)

```
./lda2bin < FORTHS.LDA > forths.bin
   Memory range is [001000, 016433]
   Execution begins at 001000
./makerx > rx01_figforth-1.3.1.dsk
./view1000 < rx01_figforth-1.3.1.dsk

         0        1        2        3        4        5        6        7
 *01000  01234567 01234567 01234567 01234567 01234567 01234567 01234567 01234567
000 0__  +++++++. ....+... ...+++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++
000 1__  ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++
000 2__  ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++
000 3__  ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++

000 4__  ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++++++++ ++...++. ........
000 5__  ........ ........ ........ ........ ........ ........ ........ ........
000 6__  ........ ........ ........ ........ ........ ........ ........ ........
000 7__  ........ ........ ........ ........ ........ ........ .....
```

$016433_8 < 017400_8$ ✔

RT-11 Directory Check

```
.DIR DX0:

FORTH .DAT    140  21-Jan-80       FORTH .MAC    146  21-Jan-80
 2 Files, 286 Blocks
 194 Free blocks

.DIR/VOL DX0:

 Volume ID: FORTH-1.3.1
 Owner    : F.I.G.
FORTH .DAT    140  21-Jan-80       FORTH .MAC    146  21-Jan-80
 2 Files, 286 Blocks
 194 Free blocks

.
```

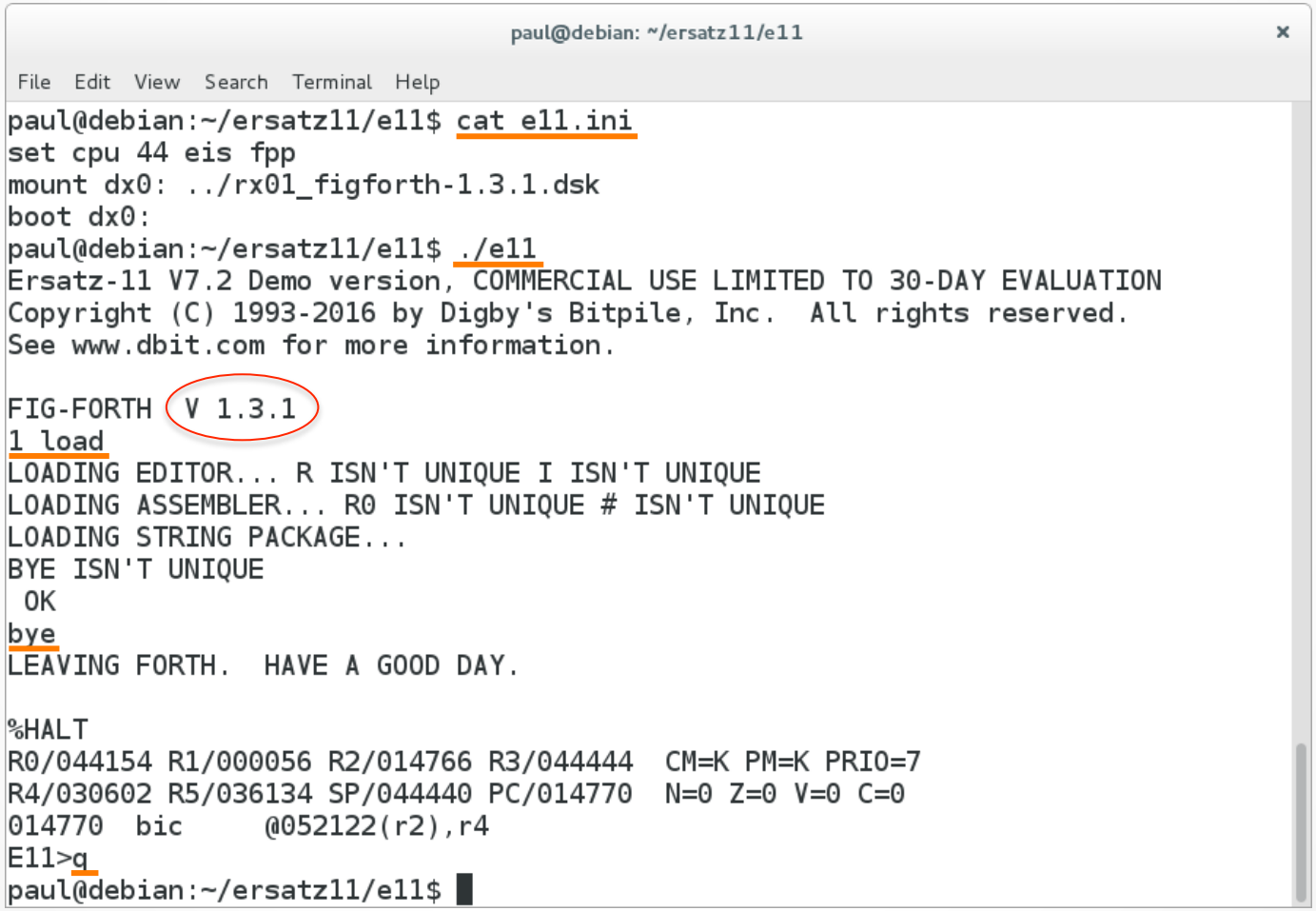
The Moment of Truth…

Missed It by >THAT< Much

```
paul@debian: ~/ersatz11/e11
File  Edit  View  Search  Terminal  Help
paul@debian:~/ersatz11/e11$ cat e11.ini
set cpu 44 eis fpp
mount dx0: ../rx01-standalone.dsk
boot dx0:
paul@debian:~/ersatz11/e11$ ./e11
Ersatz-11 V7.2 Demo version, COMMERCIAL USE LIMITED TO 30-DAY EVALUATION
Copyright (C) 1993-2016 by Digby's Bitpile, Inc.  All rights reserved.
See www.dbit.com for more information.

FIG-FORTH  V 1.3
1 load
LOADING EDITOR...  ISN'T UNIQUE  ISN'T UNIQUE
LOADING ASSEMBLER... R0 ISN'T UNIQUE  ISN'T UNIQUE
LOADING STRING PACKAGE...
BY ISN'T UNIQUE
 OK
bye
LEAVING FORTH.  HAVE A GOOD DAY.

%HALT
R0/044146 R1/000056 R2/014760 R3/044436  CM=K PM=K PRIO=7
R4/030574 R5/036126 SP/044432 PC/014762  N=0 Z=0 V=0 C=0
014762  bic     @052122(r2),r4
E11>q
paul@debian:~/ersatz11/e11$
```

Source: *Get Smart*

“Cleared” for Takeoff!



Acid Test I: Save Image

```
34 list
SCR # 34
  0 ( CREATE BOOTABLE IMAGE ON SCREENS 40-47.  FOR STAND-ALONE.)
  1 ( NOTE - THIS DOES NOT WRITE THE BOOT BLOCK OR THE OTHER FORTH)
  2 ( SCREENS.  IF YOU START WITH A BLANK DISK, FIRST USE THE COPY)
  3 ( PROGRAM ON SCREEN 38, AND MOVE THE COPY TO DX0. THEN EXECUTE)
  4 ( 'DECIMAL 34 LOAD'.  THE BOOT LOADER WILL ONLY HANDLE)
  5 ( IMAGES UP TO 7.9K BYTES.  THIS LEAVES SEVERAL HUNDRED)
  6 ( BYTES FOR NEW OPERATIONS, AND THESE COULD LOAD MORE.)
  7 DECIMAL    : SIZETEST 1024 8 * 256 -  HERE U< IF ." TOO BIG"
  8    QUIT THEN ;   SIZETEST   FORGET SIZETEST
  9 OCTAL      ( NEXT LINE RESETS THE START-UP TABLE.)
 10 LATEST 14 +ORIGIN !   HERE 36 +ORIGIN !  HERE 34 +ORIGIN !
 11 DECIMAL   35 LOAD   CREATE-BINARY-IMAGE  ( WRITE SYSTEM)
 12 10 LOAD 11 LOAD 12 LOAD 13 LOAD 14 LOAD 15 LOAD  ( ASSEMBLER)
 13 36 LOAD  ( WRITES BOOT LOADER AT END OF SCREEN 47)
 14 COLD  ( COLD START OF NEW SYSTEM - GET RID OF ASSEMBLER ETC.)
 15
 OK
decimal 34 load R0 MSG # 4  # MSG # 4  TASK MSG # 4
FIG-FORTH  V 1.3.1
```

Acid Test II: Disk Copy

```
FIG-FORTH  V 1.3.1
38 list
SCR # 38
  0 ( DISK COPY FROM SYSTEM DISK TO DX1)
  1 DECIMAL   20000 CONSTANT C
  2 : GET 26 0 DO   C I 128 * +   OVER   I 3 * 26 MOD 1+   RTS
  3    LOOP ;
  4 : PUT 26 0 DO   C I 128 * +   OVER 77 +   I 3 * 26 MOD 1+   WTS
  5    LOOP ;
  6 : COPY 77 0 DO   I GET DROP   I PUT DROP   LOOP ;
  7
  8
  9
 10
 11
 12
 13
 14
 15
 OK
38 load  OK
copy  OK
```

Leap Frog, RX01 Style

- **26 Sectors** numbered 1 through 26
- **Sector Interleaving:** skip at least every other sector
- **2 is** a prime factor of 26; **3 is not**
- **Code:** `26 0 DO I 3 * 26 MOD 1+ LOOP`
- **Sector Sequence:**

```
1   4   7   10   13   16   19   22   25
2   5   8   11   14   17   20   23   26
3   6   9   12   15   18   21   24
```

SIMH Boot: the Easy Way

```
bash-3.2$ pdp11

PDP-11 simulator V3.9-0
sim> attach rx0 rx01_figforth-1.3.1.dsk
RX: buffering file in memory
sim> boot rx

FIG-FORTH  V 1.3.1
1 load
LOADING EDITOR... R ISN'T UNIQUE I ISN'T UNIQUE
LOADING ASSEMBLER... R0 ISN'T UNIQUE # ISN'T UNIQUE
LOADING STRING PACKAGE...
BYE ISN'T UNIQUE
 OK
```

SIMH Boot: the Hard Way

BOOT0

```
PDP-11 simulator V3.9-0
sim> d 1000 005000
sim> d 1002 012701
sim> d 1004 177170
sim> d 1006 105711
sim> d 1010 001776
sim> d 1012 012711
sim> d 1014 000003
sim> d 1016 005711
sim> d 1020 001776
sim> d 1022 100405
sim> d 1024 105711
sim> d 1026 100004
sim> d 1030 116120
sim> d 1032 000002
sim> d 1034 000770
sim> d 1036 000000
sim> d 1040 005000
sim> d 1042 000110
sim> e 1000
1000:   005000
sim> attach rx0 rx01_figforth-1.3.1.dsk
RX: buffering file in memory
sim> run 1000

FIG-FORTH  V 1.3.1
```

FORTH.MAC Changes

```
;
;  CHANGES IN V 1.3.1 (PH):
;     - CHANGED VERSION NUMBER FROM "1.3" TO "1.3.1".
;     - CHANGED CPU FROM "11" (OCTAL) TO "11." (DECIMAL).
;     - ADDED "BIC #177600,(S)" TO CLEAR PARITY BIT & UPPER
;          BYTE FOR STANDALONE TERMINAL ASCII OUTPUT.
;     - CHANGED ".WORD" TO ".BLKW" IN DSKBUF AREA SO STANDALONE
;          BINARY IMAGE CAN BE VERIFIED TO FIT IN 017400 BYTES,
;          TO GUARANTEE ROOM FOR THE 2-SECTOR BOOT LOADER.
;
```

What's Included?

PDP-11 fig-FORTH DISKETTE

In addition to the fig-FORTH system, the diskette includes an editor, Forth assembler, string package and an 80-page User's Guide with discussion and examples of Forth programming techniques.

Like all the fig-FORTH systems, this one has full length names to 31 characters, the "security package" of compile-time error checks, and alignment with the 1978 Forth International Standard.

This system runs under the RT-11 or RSX-11M operating systems, and can be modified for other environments or for stand-alone It can interface to database packages or other software, allowing interactive access and program development with systems not otherwise available interactively. The fig-FORTH model is distributed in Macro-11 source, for easy modifiability by programmers without a Forth background; the editor, assembler, and string package are in Forth source.

The complete system price is $130, including diskette and all documentation; the User's Guide separately is $20. (Ca. residents add 6% tax.) John S. James, P.O. Box 348, Berkeley, CA 94701.

FORTH DIMENSIONS I/5

Forth Dimensions, 1:5
Jan/Feb 1980

What's Next?

- Seek glossaries for: micro-Forth, Forth-77
- Add a few comments in FORTH.DAT
- Tweak RX01 contents (include BOOT sources, etc.)

Resources

- RX01 Bootable Disk Image, source files, utilities, etc.: http://www.stackosaurus.com/figforth
- RT-11v4 & v5.3 (note hobbyist license): http://simh.trailing-edge.com/software.html
- Ersatz-11 (Demo Version): http://www.dbit.com/demo.html
- PUTR: http://www.dbit.com/putr/
- Empty PDP-11 Disk Images (for system generation): http://www.dbit.com/pub/pdp11/empty/
- SIMH: http://simh.trailing-edge.com/
- Original Forth Interest Group Files: http://www.forth.org/fig-forth/contents.html

Backup

TABLE, in MACRO-11

```
TABLE:  .BYTE   17, 27,  7  ; TRACK 17 SECTORS 27- 5
        .BYTE   17, 10, 10  ; TRACK 17 SECTORS 10- 6
        .BYTE   20, 15, 15  ; TRACK 20 SECTORS 15-13
        .BYTE   20, 16, 16  ; TRACK 20 SECTORS 16-14
        .BYTE   21, 23, 23  ; TRACK 21 SECTORS 23-21
        .BYTE   21, 24, 26  ; TRACK 21 SECTOR  24
        .BYTE    0          ; END OF TABLE MARK
        .EVEN               ; SKIP TO EVEN ADDRESS
```

```
: TABLE,
  17    27 2,  7
  17 2, 10    10 2,
  20    15 2, 15
  20 2, 16    16 2,
  21    23 2, 23
  21 2, 24    26 2,
   0     0 2,  ;
```

Screen 34: Forth Image

```
SCR # 34
  0 ( CREATE BOOTABLE IMAGE ON SCREENS 40-47.  FOR STAND-ALONE.)
  1 ( NOTE - THIS DOES NOT WRITE THE BOOT BLOCK OR THE OTHER FORTH)
  2 ( SCREENS.  IF YOU START WITH A BLANK DISK, FIRST USE THE COPY)
  3 ( PROGRAM ON SCREEN 38, AND MOVE THE COPY TO DX0. THEN EXECUTE)
  4 ( 'DECIMAL 34 LOAD'.  THE BOOT LOADER WILL ONLY HANDLE)
  5 ( IMAGES UP TO 7.9K BYTES.  THIS LEAVES SEVERAL HUNDRED)
  6 ( BYTES FOR NEW OPERATIONS, AND THESE COULD LOAD MORE.)
  7 DECIMAL   : SIZETEST 1024 8 * 256 -  HERE U< IF ." TOO BIG"
  8    QUIT THEN ;   SIZETEST   FORGET SIZETEST
  9 OCTAL      ( NEXT LINE RESETS THE START-UP TABLE.)
 10 LATEST 14 +ORIGIN !   HERE 36 +ORIGIN !  HERE 34 +ORIGIN !
 11 DECIMAL   35 LOAD   CREATE-BINARY-IMAGE   ( WRITE SYSTEM)
 12 10 LOAD 11 LOAD 12 LOAD 13 LOAD 14 LOAD 15 LOAD   ( ASSEMBLER)
 13 36 LOAD   ( WRITES BOOT LOADER AT END OF SCREEN 47)
 14 COLD   ( COLD START OF NEW SYSTEM - GET RID OF ASSEMBLER ETC.)
 15
 OK
```

Screen 35: Forth Image

```
SCR # 35
  0 ( CREATE A BINARY IMAGE ON SCREENS 40 - 47  )
  1 ( START AT ZERO)
  2 : CREATE-BINARY-IMAGE  48 40 DO
  3    I 40 - 1024 *  ( ADDRESS TO MOVE FROM)
  4    I BLOCK ( ADDRESS TO MOVE TO)
  5    1024 CMOVE     UPDATE    LOOP    FLUSH ;
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15
 OK
```

Screen 36: Boot Loader

```
SCR # 36
  0 ( CREATE BOOT LOADER.  NOTE - DOES NOT WRITE BOOT BLOCK)
  1 ASSEMBLER DEFINITIONS OCTAL
  2 : INIT, 1000 # R0 MOV,   00000 # R1 MOV,
  3    177170 # R4 MOV,   200 # R3 MOV, ;
  4 : ?TERM, R1 () TSTB,   LE IF, 1000 @# JMP, ENDIF, ;
  5 : WAITT, BEGIN, R3 R4 () BIT, NE UNTIL, ;
  6 : WAITD, BEGIN, 40 # R4 () BIT, NE UNTIL, ;
  7 : ?ERR, R4 () TST, LE IF, HALT, ENDIF, ;
  8 : BLOOP, R3 R2 MOV,
  9    BEGIN, WAITT, 2 R4 I) R0 )+ MOVB,   R2 DEC,   EQ UNTIL, ;
 10 : NEXTTAB, 1 R1 I) R5 MOVB,   R5 INC, R5 INC,
 11    R5 32 # CMP, GT IF, 32 # R5 SUB, THEN,
 12    R5 1 R1 I) MOVB,   1 R1 I) 2 R1 I) CMPB,
 13    EQ IF, 3 # R1 ADD, ENDIF,  ;
 14
 15    DECIMAL 37 LOAD
 OK
```

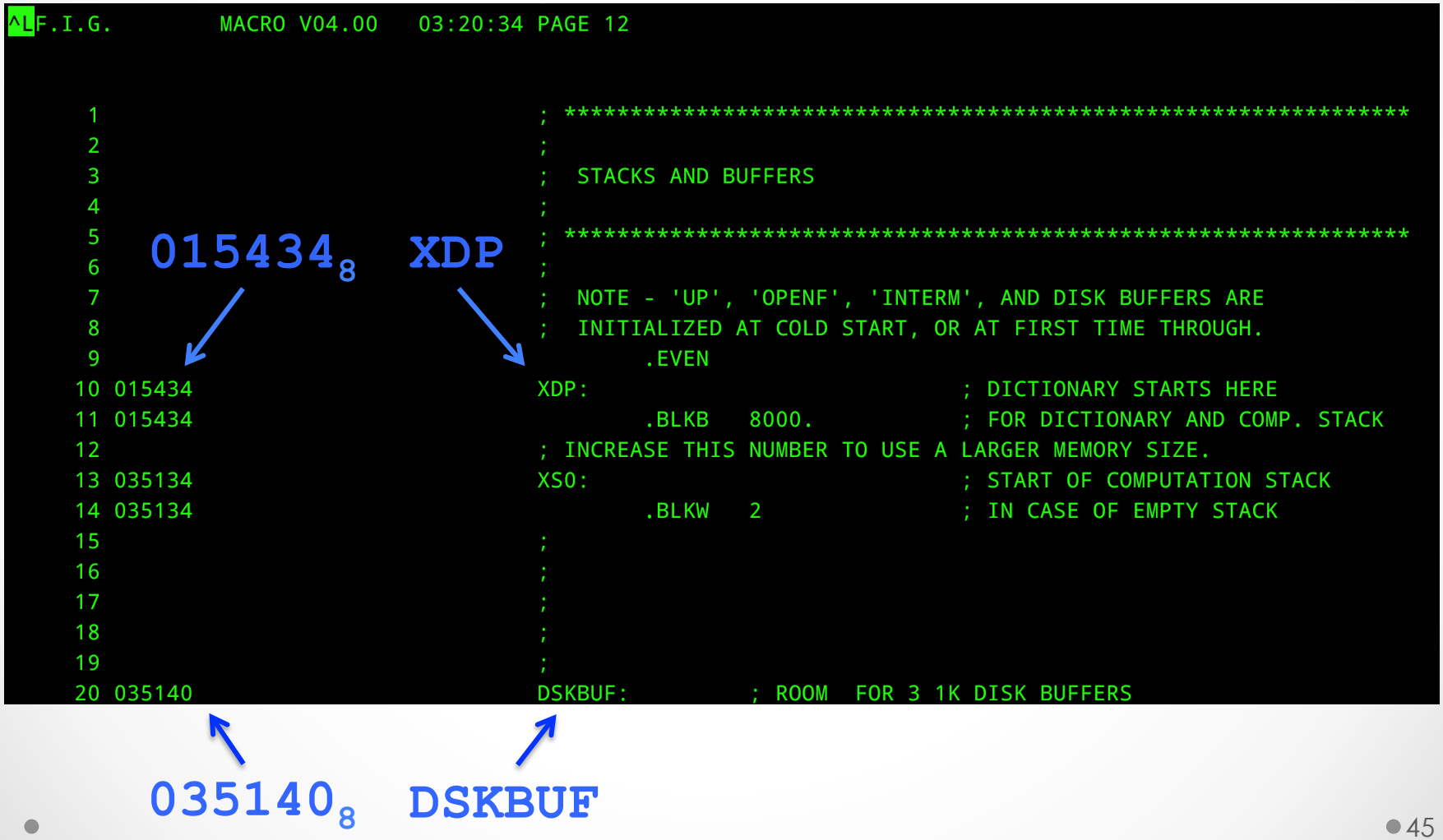
Screen 37: Boot Loader

```
SCR # 37
  0 ( CREATE BOOT LOADER, CONT.)       OCTAL
  1 : TRACK, R1 () R5 MOVB, R5 2 R4 I) MOV, ;
  2 : SECTOR, 1 R1 I) R5 MOVB,   R5 2 R4 I) MOV, ;
  3 : MAINL, BEGIN, ?TERM, 7 # R4 () MOV, WAITT, SECTOR, WAITT,
  4    TRACK, WAITD, ?ERR,  3 # R4 () MOV,  BLOOP, NEXTTAB,
  5    400 UNTIL, ;
  6 : 2, 400 * + , ;
  7 : TABLE,   17 27 2,   7 17 2,   10 10 2,
  8    20 15 2,   15 20 2,   16 16 2,
  9    21 23 2,   23 21 2,   24 26 2,   0 0 2,  ;
 10
 11 CODE BOOT 35000 JMP, C;
 12 : TASK ;
 13 35000 DP ! HERE 6 + INIT, WAITD, MAINL, HERE SWAP ! TABLE,
 14 FORGET TASK
 15 17572 35006 !   35000 21 26 WTS  35200 21 30 WTS
 OK
```

Screen 38: RX01 Copy

```
SCR # 38
  0 ( DISK COPY FROM SYSTEM DISK TO DX1)
  1 DECIMAL   20000 CONSTANT C
  2 : GET 26 0 DO   C I 128 * +   OVER   I 3 * 26 MOD 1+   RTS
  3    LOOP ;
  4 : PUT 26 0 DO   C I 128 * +   OVER 77 +   I 3 * 26 MOD 1+   WTS
  5    LOOP ;
  6 : COPY 77 0 DO   I GET DROP   I PUT DROP   LOOP ;
  7
  8
  9
 10
 11
 12
 13
 14
 15
 OK
```

FORTHS.LST


^LF.I.G. MACRO V04.00 03:20:34 PAGE 12
; **
; STACKS AND BUFFERS
; **
; NOTE - 'UP', 'OPENF', 'INTERM', AND DISK BUFFERS ARE
; INITIALIZED AT COLD START, OR AT FIRST TIME THROUGH.
.EVEN
015434 XDP: ; DICTIONARY STARTS HERE
015434 .BLKB 8000. ; FOR DICTIONARY AND COMP. STACK
; INCREASE THIS NUMBER TO USE A LARGER MEMORY SIZE.
035134 XS0: ; START OF COMPUTATION STACK
035134 .BLKW 2 ; IN CASE OF EMPTY STACK
035140 DSKBUF: ; ROOM FOR 3 1K DISK BUFFERS
015434_8 XDP
035140_8 DSKBUF

Cold Start Saves the Day

```
        HEAD    204,COLD,240,COLD                       ; ***** COLD
CENT:                           ; COLD START ENTRY POINT
        MOV     ORIGIN+14,FORTH+6 ; SET 'FORTH' VOCABULARY FROM STARTUP TABLE
        MOV     ORIGIN+20,U     ; INITIALIZE USER POINTER
;  NOTE - FOR SMALLER STAND-ALONE BOOT, INITIALIZE AREAS IN
;  HIGH MEMORY WHICH MUST BE INITIALIZED.
;  CLEAR DISK BUFFERS ON FIRST TIME THROUGH
        MOV     ORIGIN+42,R0    ; 'FIRST' - BEGINNING OF DISK BUFFERS
        MOV     ORIGIN+44,R1    ; 'LIMIT' - JUST BEYOND DISK BUFFERS
1$:     CLR     (R0)+
        CMP     R0,R1
        BLT     1$
```

boot1.bin Dump

```
od -t o2 boot1.bin
0000000    000240  000416  000036  000343  000036  000344  000030  000340
0000020    000070  000340  000036  000345  000001  000002  000036  000000
0000040    012700  017400  012706  001000  012704  177170  000004  000414
0000060    000036  000341  000036  000342  005714  001776  100741  000002
0000100    000076  000340  000076  000340  012705  177172  012701  000026
0000120    012714  000007  000004  010115  000004  012715  000021  000004
0000140    012714  000003  000004  111520  000004  105714  100774  022701
0000160    000026  001402  000167  017210  012701  000030  000751
0000176
```

boot2.bin Dump

```
od -t o2 boot2.bin
0000000    012700  001000  012701  017572  012704  177170  012703  000200
0000020    032714  000040  001775  105711  003002  000137  001000  012714
0000040    000007  030314  001776  116105  000001  010564  000002  030314
0000060    001776  111105  010564  000002  032714  000040  001775  005714
0000100    003001  000000  012714  000003  010302  030314  001776  116420
0000120    000002  005302  001372  116105  000001  005205  005205  020527
0000140    000032  003402  162705  000032  110561  000001  126161  000001
0000160    000002  001002  062701  000003  000716  013417  007407  004010
0000200    006420  010015  007016  011421  010423  013024  000000
0000215
```