Installing K2FDOS in ROM on SBC-85

K2FDOS consists of five parts: Boot, Sysgen, Format, PIP, and FDOS. When running FDOS in ROM, Boot and Sysgen are not really needed, but they are included in the package for possible future use. A special utility program, mbmutil is also included, which can be used to read and display data from the MBM by track and sector, among other things. Since PIP is used frequently it can be run directly in ROM from FDOS. Format is used much less often, and can be run from mbmutil. Alternatively, both PIP and Format can be run from FDOS like any other program, if you've saved copies on the MBM.

The version of K2FDOS in this package is configured and assembled for an SBC-85 system with 8KB ROM at address 2000H, 8KB RAM at address 8000H, and 256 bytes at A000. You also need the Dunfield monitor, version 1.4, in ROM at address 0000H. The Dunfield monitor is required because FDOS makes use of interrupts, specifically RST 2, using the monitor's mechanism for relocating the vectors. For this reason, it is not possible to run programs that use FDOS functions directly from the monitor. They need to be run from FDOS. And FDOS probably won't work with a different monitor without modification.

When you unpack the fdos tar file you will have a variety of files with extensions like .asm, .hex, .asc, .lst, and .bin. Depending on your PROM programmer, you will probably want the .hex or .bin files. In ROM, FDOS starts at 2000, PIP at 2D00, and MBMUTIL at 3500. Program an 8KB EPROM (2764) as follows:

k2fdos at offset 0	(2000 -> 0000)
pip at offset 0D00H	(2D00 -> 0D00)
mbmutil at offset 1500H	(3500 -> 1500)

Those offsets are determined by subtracting 2000H (the EPROM address) from the ORG address of each program. To help verify that the programmer has the right data in the right places, here are the first 64 bytes of each program as they should appear in the programmer.

```
FDOS:
00000000  C3 11 20 05 9D EF 9E 77  9F 10 9D AF 9E 13 9D 2F  |.. ....w......./|
00000010  9E C3 A4 20 C3 D3 27 C3  D3 27 C3 12 28 C3 12 28  |... ..'..'..(..(|
00000020  C3 CA 28 C3 AB 29 C3 AB  20 C3 E0 28 C3 FE 22 C3  |..(..).. ..(..".|
00000030  07 23 C3 5B 24 C3 39 24  C3 36 25 C3 92 24 C3 CB  |.#.[$.9$.6%..$..|

PIP:
00000D00  21 8C 2E D7 15 D7 08 01  26 04 21 BF 98 AF 77 23  |!.......&.!...w#|
00000D10  0B 78 B1 C2 0E 2D 21 A2  2E D7 15 D7 17 D7 16 CA  |.x...-!.........|
00000D20  1D 2D FA 05 2D FE 3F CA  5D 2D 47 D7 16 4F D7 16  |.-..-.?.]-G..O..|
00000D30  57 21 65 2D 1E 0B 7E 23  B8 C2 4D 2D 7E 23 B9 C2  |W!e-..~#..M-~#..|

MBMUTIL:
00001500  21 F4 37 D7 15 D7 17 D7  1D DA 00 35 7D FE FF CA  |!.7........5}...|
00001510  2A 35 FE 01 DA 1C 35 FE  0D D2 00 35 3D 5F 16 00  |*5....5....5=_..|
00001520  21 E5 36 19 19 5E 23 56  EB E9 D7 07 D7 81 4C D7  |!.6..^#V......L.|
00001530  1E 4D D7 1F CD 4A 37 D7  81 0E 88 C3 1B 37 D7 82  |.M...J7......7..|
```

After installing the EPROM in the ZIF socket of the CPU board and powering up, you should be able to run FDOS at address 2000H. Following is a sample session, showing available FDOS and mbmutil commands, and finally returning to the monitor (inputs in bold):

```
     MON85 v1.4

(c)1979-2007 D.Dunfield
2020-Used with permission for SBC-85


C> G 2000

K2 FDOS VERSION 1.6
DATE? 09-MAR-23
-?

  S <filename>: Save a file    R <filename>: Run a program
  E: Exit to monitor           J <address>: Jump to address
  U: Toggle uppercase flag     M: Run the MBM Utility from ROM
  P: Run PIP from ROM          ?: Display this help
-M
FDOS/MODCAL TEST
CHOOSE ONE (HEX #S):
  1: DISPLAY WRTFILE DATA     2: DISPLAY REDFILE DATA
  3: TEST TS2BT               4: DISPLAY FDOS BITMAP DATA
  5: LOAD FDOS FROM MBM       6: TEST RNDIN
  7: FORMAT MBM FOR FDOS      8: XFER FDOS TO MBM

                FF: RETURN TO FDOS
TEST#: FF
-E

MON85 v1.4

(c)1979-2007 D.Dunfield
2020-Used with permission for SBC-85

C>
```

Installing And Using TEA

Both TEA and ASM1 in this package have been assembled to run at address 8000H. After unpacking the tar file, you'll see files with extensions like .asm, .asc, .hex, .lst, and .bin. In order to use these without modification you'll need an SBC-85 system with 8KB of RAM at 8000H, and a monitor of some kind, preferably the Dunfield monitor, at 0000H. There also needs to be some RAM between A080H and A0FFH.

The Dunfield monitor can read Intel Hex files, but other monitors may require some other method of getting the programs into RAM at the correct address. The following example shows the steps used to get TEA into RAM, starting from the Dunfield monitor prompt:

```
C> l
(at this point do whatever you need to do to get your terminal program
to send the tea-mbm.hex file)
:C>
```

TEA is now in RAM addresses 8000H through 8F92H (you can find the last address used by looking for the "END" statement in the tea-mbm.lst file). Here are the first 64 bytes as they should appear in RAM:

```
00008000  21 90 A0 22 80 A0 21 E0  A0 22 82 A0 AF 32 84 A0  |!.."..!.."...2..|
00008010  32 85 A0 32 E6 8E 32 E7  8E 31 92 8F 2A 86 A0 EB  |2..2..2..1..*...|
00008020  1B 21 3D 8B CD DC 80 EB  36 0D 23 22 86 A0 2A 8A  |.!=.....6.#"..*.|
00008030  A0 EB 21 51 8B CD DC 80  2A 86 A0 7D 93 7C 9A D2  |..!Q....*..}.|..|
```

Saving files with FDOS is done this way:

```
C> G 2000
-S TEAMBM
*L 8000,8F92
*G 8000
-
```

Some things to be aware of are that the spaces after the FDOS "S", "L", and "G" sub-commands are required. If you leave them out it won't cause an error, but the results will be disappointing. FDOS does not allow overwriting files, and if you try you will get a cryptic "ERROR 10". In that case either use a different name or use PIP to delete or rename the file. Finally, avoid saving an executable file after you've already run it. There may be areas of RAM that the program has initialized or changed and the saved file could include those changes. Instead, only save executable files after loading or assembling them in RAM, and before running them. To save a data file, finish with "Q" instead of "G ####". The K2FDOS book, available on archive.org, has more detail about using FDOS, PIP, and more.

When saving TEA and ASM1 to MBM, you can use any names you wish, as long as they fit the FDOS 6.3 format. However, both TEA and ASM1 are currently written to smoothly interact, which means you will need to use the names they expect: TEAMBM and ASM1, with no extensions.

Once a copy of TEA is saved on the MBM, you can run it from FDOS:

```
-R TEAMBM

INITIAL ADDRESS? H9000
FINAL ADDRESS? H98BF
SOURCE CURRENTLY IN MEMORY? N
C=?
TEA COMMANDS:
  A: APPEND (^C ENDS)    B: BOOT ASSEMBLED PROG.
  C: CHANGE (^C ENDS)    D: DELETE LINES
  E: EXCHANGE (^E ENDS)  H: HEX NUMBERS
  I: INSERT (^C ENDS)    L: LIST LINES
  M: MEMORY LIMITS       N: NEXT SEARCH ITEM
  O: OCTAL NUMBERS       P: PUNCH OUTPUT
  Q: QUERY MEMORY USED   R: READER INPUT
  S: SEARCH (^S ENDS)    U: UC/LC TOGGLE
  X: ASSEMBLE  (TEA)     V: ASSEMBLE (ASM1)
  Z: RETURN TO MONITOR   ?: THIS HELP

C=
```

The initial and final addresses are chosen according to whatever RAM is installed in your system. In this case, there is RAM from 8000H to 9FFFH, with approximately the first half of that taken up by TEA itself, and a chunk at the end used by FDOS. TEA always asks for those addresses, but if you've already set them you can just hit "enter" and it will use the previous values.

This is a line editor, so you normally work on one line at a time. It was also written for Teletypes that print on paper, so some behavior that may seem strange for a video display makes more sense if you think about a Teletype printing on paper.

Most of the operations, like Append, Insert, and Change, are fairly intuitive, but Search and Exchange may require some explanation. You use ^S to terminate search strings, so you'll have to configure your terminal program to ignore XON/XOFF. Once TEA has found a matching pattern that you wish to edit, use "E=" followed by the replacement pattern, terminated with ^E. Now you can continue searching with "N" if you wish, and when you find another match that you want to modify, use "E" without the equal sign, unless you want to use a different replacement string. It sounds complicated, but once you've done it a few times it's not too bad.

Editing while typing is limited to backspace to delete the previous character. The deleted character will be echoed along with "\" (a Teletype printer cannot do an actual backspace). You can use ^U to cancel the entire line and start over. And ^R can be used to retype the line so far. A lot of backspaces can get confusing, so retyping the line is helpful. These line editing functions are common for most of the programs, but there are probably still a few places where they haven't yet been implemented.

You will see in the list of commands two that are named Reader and Punch. In this version of TEA those are actually MBM read and write. In order to read a file you will need to know it's name because TEA doesn't have a directory list function - use PIP for that. After entering the name of an existing file, TEA will wait for you to select either Insert or Append. After reading in the file it will respond with the new memory usage and last line number.

To save a file, use the Punch command. It will prompt for a filename. Since FDOS doesn't allow overwriting files, if you supply an existing filename you will be prompted with a choice of renaming the existing file or deleting it.

Installing And Using ASM1

Load ASM1 into RAM using the same method as with TEA. Here are the first 64 bytes of ASM1 as they should appear in RAM:

```
00008000  CD 85 81 11 32 8D 21 95  8C CD 67 81 21 B2 8C CD  |....2.!...g.!...|
00008010  67 81 21 32 8D 7E FE 4D  C2 21 80 32 1A 8D C3 54  |g.!2.~.M.!.2...T|
00008020  80 FE 44 C2 4C 80 32 1E  8D E5 21 83 8C 11 22 8D  |..D.L.2...!...".|
00008030  CD 67 81 21 22 8D CD 57  87 2A 0B 8D 24 22 07 8D  |.g.!"..W.*..$"..|
```

Saving an executable copy of ASM1 on the MBM is similar to what has already been done for TEA (once loaded into RAM):

```
C> G 2000
-S ASM1
*L 8000,8E51
*G 8000
-
```

This is a fairly basic assembler. Pseudo ops are ORG, EQU, DS, DW, DB, INCLUDE, and END (END is optional). Labels cannot be longer than six characters and always end with ":<tab>". Comments begin with ";". Currently the only functional math operator is "+". Others are allowed but do nothing. ASM1 is designed to be run from TEA, and uses data from TEA to locate source in RAM. When started, it prompts for source location, either memory or disk. If disk, it will prompt for a filename, and like TEA, it expects you to already know the name. The next step is to specify outputs: memory, disk, printer, or console. Console output will assemble and produce a listing on the console. Assembling to memory will produce a binary image at the source's ORG address with no listing. Output to disk writes the binary result to the filename you supply, with no listing. Choosing printer output currently does nothing since no code to drive a printer has been included. To summarize, for a binary output choose memory or disk. For a listing choose console or, if enabled, printer.

Example ASM1 session, starting from TEA:

```
C=L
0001  COU:    EQU     03D7H
0002  RESTRT: EQU     07D7H
0003
0004          ORG     5000H
0005
0006  S:      MVI     C,"%"
0007          DW      COU
0008          DW      RESTRT
0009          END
0010
C=VSOURCE FROM M)EMORY, D)ISK? M
OUTPUT TO M)EMORY, D)ISK, P)RINTER, C)ONSOLE? C
03D7              0001  COU:    EQU     03D7H
07D7              0002  RESTRT: EQU     07D7H
                  0003
5000              0004          ORG     5000H
                  0005
5000 0E 25        0006  S:      MVI     C,"%"
5002 D7 03        0007          DW      COU
5004 D7 07        0008          DW      RESTRT

INITIAL ADDRESS?
```

With an error:

```
C=VSOURCE FROM M)EMORY, D)ISK? M
OUTPUT TO M)EMORY, D)ISK, P)RINTER, C)ONSOLE? C
03D7              0001  COU:    EQU     03D7H
07D7              0002  RESTRT: EQU     07D7H
                  0003
5000              0004          ORG     5000H
                  0005
SYNTAX ERROR AT LINE # 0006-> "%"
5000 0E 00        0006  S:      MVI     C, "%"
5002 D7 03        0007          DW      COU
5004 D7 07        0008          DW      RESTRT

INITIAL ADDRESS?
```

Notice that it's the space after the comma that's the problem, as indicated by the -> arrow. And there's no symbol table output yet.

Glossary

Boot: The boot program for FDOS. It reads the first two tracks from the MBM, which is where FDOS is normally stored.

FDOS: The main component of the K2FDOS system. It provides a collection of disk and other I/O related functions that can be easily utilized by other programs.

Format: The program that's used to initialize a storage device, in this case the MBM. It simply fills each FDOS block with zero, except for two bytes that always contain the track and sector numbers. Since there are two sectors, or MBM pages, per block, the sector numbers will always be even.

K2FDOS: The collection of five programs including Boot, Format, Sysgen, PIP, and FDOS.

MBM: Magnetic Bubble Memory. Each module organizes data into 2048 pages of 68 bytes each (64 bytes when using error correction). FDOS treats each page as a disk sector.

MBMUTIL: Utility program that provides some additional functions, like reading any FDOS block by specifying its track and sector numbers, formatting the MBM, etc.

PIP: PIP provides a convenient interface for performing common file related tasks, such as delete, rename, copy, list, etc.

Sysgen: This program copies the executable binary image of FDOS from RAM to the first two tracks of the MBM.

Summary of System Requirements

Hardware configuration:

1. 8K ROM at 0000 with Dunfield monitor version 1.4
2. 8K ROM at 2000
3. 8K RAM at 8000
4. 256 bytes RAM at A000

Memory, both ROM and RAM used by each program:

PROGRAM	ROM	RAM
FDOS	2000-2C7B	98BF-9FFF
PIP	2D00-34D1	98BF-9CE5
MBMUTIL	3500-3A48	98BF-9947
TEA		8000-8F92
ASM1		8000-8E52

In addition, TEA and ASM1 share some RAM from A080-A0FF. FDOS initializes A000 which is used by the monitor to redirect interrupt vectors. The apparent RAM overlap between FDOS, PIP, and MBMUTIL is intentional.

You could probably use a 128K EPROM at 0000 holding the monitor, FDOS, PIP, and MBMUTIL, saving the ZIF socket for something else. But this is the way I have my system configured.

Location of ORG Statements in Source Files

FDOS code	k2fdos.asm line 126	`org 02000H`
FDOS data	k2fdos.asm line 1863	`org ramtop-(031ah+0426h)`
PIP code	pip.asm line 54	`ORG 02D00H`
PIP data	pip.asm line 870	`ORG RAMUSR`
MBMUTIL code	mbmutil.asm line 31	`ORG 03500H`
MBMUTIL data	mbmutil.asm line 611	`ORG RAMUSR`
TEA code	tea-mbm.asm line 92	`ORG 08000H`
TEA data	tea-mbm.asm lines 21-31	`EQUates`
ASM1 code	asm1.asm line 41	`ORG 08000H`
ASM1 data	asm1.asm lines 15-17	`EQUates`

RAMTOP and RAMUSR are defined in fdosincl.asm:

```
RAMTOP          EQU     9FFFH
RAMUSR          EQU     RAMTOP-(031AH+0426H)
```

FDOS uses 031A bytes, and PIP/MBMUTIL use 0426 bytes.