FLEX™ NEWSLETTER NO. 4
November 1980

---

Copyright (c) 1980 by Technical Systems Consultants, Inc.
P.O. Box 2570, West Lafayette, Indiana 47906

---

Things are going strong in the world of FLEX. Our "General Use" versions have been out for quite some time now and permit a whole new base of users to run FLEX software. We see a lot of interest and use of FLEX in Europe where there seems to be a higher percentage of 68XX systems than in the US. Many hardware manufacturers are also quite pleased with the general versions of FLEX since it permits their users to quickly get on-line with a complete array of development software and application packages. We've got lots of good info for you this time, so let's get right to it.

## 1) FLEX News

---

Since the last newsletter we have begun shipment of a version of 6809 FLEX for operation on a Motorola 6809 EXORciser with EXORdisk II or III. Like the 6800 version, it requires no hardware mods (with the possible exception of memory readdressing) and uses the standard 6809 MDOS ROM boot for booting. Editor and assembler are included and all sells for $150.00.

We have received several queries about a version of 6809 FLEX for the Motorola EXORset 30 computer system. We would love to oblige those users, but unfortunately the hardware will not support FLEX. The main stumbling block is the disk controller used in the EXORset. It utilizes Motorola's 6843 floppy disk controller chip which can only support 128 byte sectors, not the 256 byte sectors which FLEX requires. Therefore FLEX cannot be brought up without significant hardware changes.

Gimix Inc, from Chicago have licensed 6809 FLEX and are now shipping versions for their new floppy disk controllers. They have one controller that supports single-density, double-sided 8 or 5 inch drives (can be mixed) and another supporting 5 inch double-density drives. A new DMA controller is in the works that will support either 8 or 5 inch floppies. You can get 6809 FLEX directly from Gimix for any of these configurations. It runs with either GMXBUG-09 or SBUG-E. Contact Gimix for details.

Smoke Signal Broadcasting hardware users are getting a version of 6809 FLEX. Smoke Signal Broadcasting themselves will be marketing a version which operates with all Smoke Signal hardware, ie. their 6809 CPU card and their DCB-4 disk controller card. That version includes a ROM containing FLEX drivers which is plugged into the CPU card. The user can then run either operating system (6809 DOS or FLEX) without any

hardware mods. If you are interested, or know someone who is, contact Smoke Signal Broadcasting directly.

Ripley Computers, 126 N. Main, Souderton, PA 18964, is offering a set of patches which can be overlayed to a standard version of 6809 FLEX to create a version which operates on a Smoke Signal system using an existing 6800 disk controller board. We have not seen this package run, but it sounds quite interesting and you might want to check it out. They call it "Power Patch F1 - FLEX".

We now have a double-sided, double-density version of 6800 FLEX for operation with SWTPc's DMF-2 controller board. There is a slight problem in that the DISKBUG ROM from SWTPc for 8 inch 6800 system does not have a double-density compatible boot. We suggest a couple of alternatives with the package. Once you have gotten around the booting problem, the double-density 6800 FLEX works great.

## 2) New Products

---

Before describing what new products are in the works, we wish to draw your attention to one new FLEX package which we have just completed, namely the 6800 Diagnostics Package. A brief description of that package follows later in the newsletter.

As most of you know by now, we are shipping copies of our 6809 UniFLEX™ Operating System. As of this printing the only versions available are for the SWTPc S/09 computer and for a Gimix 6809 system with a SWTPc DMAF-2 disk controller. You will have to see UniFLEX run to appreciate it. To begin with, the disk transfer rates for floppy disk are on the order of 24K per second for both reading and writing. That's on the order of eight times the speed of FLEX for reading and sixteen times as fast for writing if you have verify on under FLEX. We think that is quite amazing! With a hard disk, our disk transfer rates are not much below that of a PDP-11/70! UniFLEX is a full multi-user system, where users "log-in" with names and confidential passwords. It is also multi-tasking. For example, it would be simple for one user logged onto the system to start an assembly "in the background" and then begin editing a file while that assembly was running. In fact, he could start several assemblies and have them all running simultaneously in the background while doing something else in the foreground. UniFLEX is the only currently available microcomputer operating system which supports "swapping". "Swapping" refers to the process of temporarily copying all of a task's memory to a disk so that another task can use the memory for awhile, and then copying all of the original task's memory back from disk for continued running. With a fast enough disk for swapping (a hard disk), that procedure does not slow down operation too much and permits you to run more jobs than will fit in memory at one time. As of this writing we have BASIC (mostly compatible with our FLEX Extended BASIC), a BASIC Precompiler, and a Text Processor all available for operation under UniFLEX. An editor and assembler are included in the price of UniFLEX. For more information, contact us directly.

Other projects which are in the works and which will hopefully be available soon are our PASCAL and C compilers and a Relocating Assembler and Linking Loader. The compilers will be available only for 6809 but we hope to have the assembler available for both 6809 and 6800. The PASCAL compiler will generate assembler code and follows the Jensen and Wirth description of PASCAL very closely. The C compiler also generates assembler code and it fully complies with the description in Kernighan and Ritchie's book, "The C Programming Language".

## 3) Patches and Fixes

---

### a) ALTERING THE NUMBER OF TRACKS IN 6800 FLEX 2.0 NEWDISK

We printed this last time, but left out one change. This shows how to alter the number of tracks which are formatted onto a disk via the NEWDISK command. These changes should be made in the NEWDISK code ONLY if the old values listed here match those in your particular version. All values are hexadecimal except for the one equation which has a 10 decimal as marked.

Change the following bytes:

| | | | | | |
|---|---|---|---|---|---|
| 1) | A1EC | from | 23 | to | MAXTRK |
| 2) | A245 | from | 23 | to | MAXTRK |
| 3) | A286 | from | 01 54 | to | (MAXTRK-1)*10 [10 decimal] |
| 4) | A28B | from | 22 | to | MAXTRK-1 |
| 5) | A2EA | from | 23 | to | MAXTRK |
| 6) | A3BF | from | 22 | to | MAXTRK-1 |

Where MAXTRK is the maximum number of tracks on the disk.

For example, a 40-track Wangco should have these values:

| | | | | | |
|---|---|---|---|---|---|
| 1) | A1EC | from | 23 | to | 28 |
| 2) | A245 | from | 23 | to | 28 |
| 3) | A286 | from | 01 54 | to | 01 86 |
| 4) | A28B | from | 22 | to | 27 |
| 5) | A2EA | from | 23 | to | 28 |
| 6) | A3BF | from | 22 | to | 27 |

### b) 6809 FLEX PRINTER SPOOLER PATCH

All versions of 6809 FLEX (not just ours) up to and including our version number 3.00 have an intermittent problem in the printer spooler code. If you are not using printer spooling, this will never present a problem to you. If you do use printer spooling, you may wish to make the following patch to prevent the spooler from intermittently "hanging up". The problem is caused by an interrupt conflict. It is easily fixed with the following patches:

at $C700 change from 7E C7 23 to 7E C7 21
at $C71F change from 6E 9F D3 E7 to 3B 12 1A 10

## 4) The "Wilton Hart" Fix for FLEX 2.0

Over the past couple of years we have received several reports of 6800 FLEX 2.0 generating some error which caused zero free sectors to be left on a disk. We explained in an earlier newsletter that the "zero free sectors left" was brought about by a disk error when writing a file, but were unable to explain why there were errors except to say that the user was simply experiencing disk hardware problems. Wilton Hart from Oregon was not satisfied with that explanation and proceeded to investigate the problem. He developed the theory that the errors were occuring because the MF-68 turned off the disk drives and that they would not always get up to proper speed before FLEX 2.0 attempted to read or write. The disk drivers supplied in FLEX 2.0 do not specifically check for a drive ready condition, they just assume that if the drive is not ready an error will occur. Some older Shugart drives had overshoot on the speed when turned on. In other words, the drive would speed up until past the correct speed and then settle down to the correct speed. On reading this should cause no problem, you would just receive an error. It is possible, however, that during writing the controller might think it was in the right spot and start writing when actually it could be writing anywhere on the disk. This could screw up the structure of the disk and eventually lead to an error. Mr. Hart's solution is to put a check in the drivers to see that the WD1771 controller chip is ready, indicating that the drives are ready. His patch is printed below. It is also printed in the September 1980 issue of '68' Micro Journal.

At this point, we are not 100% sure of the effectiveness of this fix for a couple of reasons. First of all, we never (or at least very seldom) experienced any difficulty with FLEX 2.0 producing "zero free sectors" errors. In all fairness, however, we do not use minifloppies very much. Most of our work is on 8 inch floppies. We have not had much of a chance to test out this patch, but such a test would not be very meaningful since we do not have problems with the system as is. Second, we have spoken to a few people who have tried this fix and some say it helped while a few have said it did not help. It appears to us that Mr. Hart may be correct in his theory and that his patch should help, but we would like to be more sure before implementing it into FLEX 2.0. We would appreciate hearing from those of you who try or have tried this patch. Are you experiencing problems with FLEX 2.0 as we distribute it? Does this patch help you?

```
                    * WILTON HART PATCH FOR FLEX 2.0 STARTUP DELAY

                    * THIS IS A 1 SECOND DELAY PROGRAM TO FIX FLEX 2.0.
                    * THE SHUGART SA400 DRIVE REQUIRES A 1 SECOND WAIT AFTER
                    * THE MOTOR COMES ON BEFORE THE DRIVE IS ACCESSED.
                    * THIS IS NOT INCLUDED IN THE STANDARD 5 1/4" FLEX.
                    *
8018                STATUS  EQU     $8018       DISK CONTROLLER STATUS REG.
                    *
BF7E                        ORG     $BF7E       PATCH CURRENT DRIVERS
BF7E BD BF B3               JSR     CHKWT       CHECK FOR DRIVE TURNING
                    *
BFB1                        ORG     $BFB1       THIS IS THE PATCH
BFB1                TMP     RMB     2           TEMPORARY STORAGE
BFB3 7D 80 18       CHKWT   TST     STATUS      IS 'NOT READY' BIT SET?
BFB6 2A 13                  BPL     RETURN      IF NOT, NO WAIT NEEDED
BFB8 FF BF B1               STX     TMP         SAVE X REGISTER
BFBB 36                     PSH A               SAVE A REGISTER
BFBC 86 7D                  LDA A   #$7D        LOAD FOR DELAY
BFBE CE 03 EE       LOOP2   LDX     #$03EE      LOAD FOR DELAY
BFC1 09             LOOP1   DEX                 DECREMENT INNER LOOP COUNTER
BFC2 26 FD                  BNE     LOOP1       LOOP TIL X=0
BFC4 4A                     DEC A               DECREMENT OUTER LOOP COUNTER
BFC5 26 F7                  BNE     LOOP2       LOOP TIL A=0
BFC7 32                     PUL A               RESTORE A REGISTER
BFC8 FE BF B1               LDX     TMP         RESTORE X REGISTER
BFCB F6 80 19       RETURN  LDA B   STATUS+1    THIS WAS REMOVED BY PATCH
BFCE 39                     RTS

                    *
                    * THIS IS THE CONFIGURATION BLOCK FOR FLEX.COR
                    * TO USE, ASSEMBLE PATCH AND APPEND TO FLEX.COR WITH:
                    *    +++APPEND,FLEX.COR,FILENAME.BIN,FLEX.SYS
                    * WHERE FILENAME IS THE CODE FROM THE ABOVE PATCH.
                    * THEN LINK THE RESULT WITH:  +++LINK,FLEX
                    *
BEA3                        ORG     $BEA3
BEA3 E1 AC          INVEC   FDB     $E1AC       INPUT CHARACTER ROUTINE
BEA5 E1 D1          OUTVEC  FDB     $E1D1       OUTPUT CHARACTER ROUTINE
BEA7 80 04          ACIA    FDB     $8004       BASE OF ACIA
BEA9 80 12          TIMER   FDB     $8012       TIMER BOARD BASE
BEAB A0 00          IRQ     FDB     $A000       IRQ VECTOR LOCATION
BEAD A0 12          SWI     FDB     $A012       SWI VECTOR LOCATION
BEAF E0 D0          MON     FDB     $E0D0       MONITOR ENTRY ADDRESS
BEB1 A0 48          PCV     FDB     $A048       MONITOR PC LOCATION
                            END     $AD00       DOS TRANSFER ADDRESS
```

## 5) Version Numbers in FLEX

Many questions have arisen from the version number technique in FLEX command files. That technique is a very simple one in which the VERSION command looks at the third byte of a binary file. That byte is considered the version number for that file. Now the important consideration is that the file whose version number you are checking must have been written with that technique in mind, ie. the third byte must be the version number. Most of our software is setup that way, but some is not. For example, the FLEX operating system itself does not have a valid version number in the third byte. Some of the first versions of BASIC did not have a true version number setup either. That is the reason you will sometimes get a strange version number like "80" or "188". Most files with ".CMD" extensions should have valid version numbers like "2" or "14".

Another source of confusion is the "VER" command supplied by SWTPc in their version of FLEX. SWTPc has devised their own method of adding a version number to a file which differs from ours. Thus, running their VER command will not give you the version number that we go by. Our version number is printed via the VERSION command, theirs is printed via the VER command.

Due to the low pricing of our FLEX based software, it is not possible for us to provide automatic maintenance. Our policy remains that if you have owned a piece of software for more than 2 months, obtaining an updated copy of a program will cost you. You must return the original disk and proof of purchase along with $10 for a new version or proof of purchase and $20 for a new version with us supplying the disk. We can also not inform you when changes are made. The version number is one method of knowing if you have the most recent release. Below is a list of several of our software packages and the most current version number as of November 1, 1980.

| Program Name | 6809 Version | 6800 Version |
|---|---|---|
| Extended BASIC | 17 | 15 |
| Extended BASIC Precompiler | 4 | 2 |
| BASIC | 13 | 12 |
| BASIC Precompiler | 3 | 2 |
| Multi-User BASIC | 13 | |
| Text Editing System | 2 | |
| Assembler | 2 | |
| Sort/Merge | 2 | 2 |

## 6) Sorting Random Access Files with SORT/MERGE

We have received a number of calls about sorting FLEX random access files with our Sort/Merge package. It's really not difficult to do as we'll explain here. To make things easiest, you should obtain version #2 of sort/merge if you don't already have it. To determine what version you have, run a quick test sort and see what version number is printed in the header sent to the screen. Version #2 has a few minor bugs fixed up and has one added feature: the ability to create a random file on output. Armed with that version you are ready to sort random files.

There are two things different about specifying how the input file is to be read. First you will probably be wanting to sort on some fixed record length rather than having an end-of-record character. For example, if you have records that are 252 characters in length (one record = one full data sector), you would probably want to specify that your input records were fixed length records of length 252. The second thing you need to inform sort/merge about is the fact that the input file is binary and not textual. This is done during the additional input parameter prompting from the SORT command. Sort/merge assumes textual input files unless you specifically tell it that the input is binary. We want to sort the input file as binary so that no space expansion goes on thus screwing up record lengths.

There are three things to tell sort/merge about the output file being created. First you must declare the output file as binary instead of textual so that no space compression goes on which would mess up record lengths in the output file. Second you need to declare the output file as a random access file. This is done in the additional prompting section of the SORT utility (only in version #2 or higher). If you do not specify the output file as random, it will be output as a sequential file. Third, you will most likely want to specify no end-of-record character for the output file. By default, sort/merge outputs a carriage return at the end of each output record. Assuming we are sorting fixed length records, this would add a character to each output record and thus mess up the record lengths. In the additional prompting section of SORT you are asked for an "EOR CHARACTER FOR OUTPUT RECORDS (DEFAULT=$0D)?". Responding with an 'N' tells sort/merge to not append any character on the end of the output records.

That's all there is to it. By making sure that you properly specify these five points, sorting random files should present no problem.

## 7) Recovering From an Editor Crash

Have you ever spent an hour entering text into the FLEX text editor only to have something go wrong when you try to terminate the session and not get your file written to disk? If not, you probably will someday. It might be that you opened the disk door for some reason and that gives an error, or maybe you really get a hardware error or glitch which causes the editor to kick out with an error message and go immediately back to FLEX. It's very frustrating to lose all that work and we've got a way that some of it might be salvaged.

The FLEX text editor performs all the editing functions in memory. When you are entering text or making changes to it, that is all stored in memory. Only when you have finished editing the text in memory and tell the editor to "STOP" or to do a "NEW" command, does the current memory buffer of text get written to disk. If that write operation should fail for any reason, the editor gives an appropriate error message and immediately returns to FLEX. Nothing is done to the contents of the user memory which means the latest buffer of text is still there. We will tell you here how to recover that buffer full of text.

Let's assume we were editing a file on drive 1 and got an error when we tried to stop the edit session. First remove the disk which would have received the file from drive 1. Now insert another formatted disk which has free space on it into drive 1. By using the "JUMP" command in FLEX or by exiting FLEX and using your ROM monitor, restart execution of the editor at its warm start address. The warm start address of the 6800 FLEX editor is $0203 while the 6809 FLEX editor warm start address is $0003. When you jump to that address you should immediately receive the editor's prompt ("#"). If you try printing the contents of the memory buffer with a "PRINT!" command, you will see how much of the file you can recover. This recovery is accomplished via the "SAVE" command. When you enter it, you will be prompted for whether you wish to save to tape or disk. Respond with a "D" for disk. Now you will be asked for a filename which you should supply. At this point the editor will save the contents of the current memory buffer into a file by the name you specified and you will receive another editor prompt. DO NOT ENTER A "STOP" OR "S" COMMAND! It is crucial that you do not attempt a normal stop of the editor because it thinks it is still writing to the original disk which was in the drive and would most likely destroy the structure of the new disk you have inserted! At this point you should hit the hardware reset button on your computer to get out of the editor and reenter FLEX by jumping to FLEX's warm start or by re-booting. You now, however, have the contents of the memory buffer in a good disk file. If you were editing a large file and had already performed one or more "NEW" commands, then you may be able to still recover the text which has already been written to the new file on the original disk. By merging this text with the text you recovered from memory with perhaps portions of the original text file (if there was one), it may be possible to completely recover all your work!

## 8) 6800 Diagnostics Package

Our 6809 Diagnostics Package has been extremely well received. In a review in '68' Micro Journal, Don Williams (Publisher) said, "Of all the utility packages that TSC has ever offered (and I have practically all 6800/6809) this collection is, in my opinion, the finest they have produced." In a later review in '68' Micro Journal, Ron Anderson (FLEX User Notes Editor) says, "I would have to try very hard to find something to complain about." They are very powerful utilities and perhaps the best indication of their worth is how much we use them here. Now, in response to great demand from 6800 FLEX users, we are proud to announce availability of the same package of routines for use under 6800 FLEX.

The utilities are divided into two groups: six memory diagnostics and ten disk repair programs. The memory diagnostics are a variety of tests including walking bit tests, convergence tests, a random pattern test, and a dynamic RAM dropout test. The disk repair utilities are aimed at recovering data from or repairing the data on "structurally damaged" disks. They will get you around problems like CRC errors, crashed directories, accidentally deleted files, etc. They include routines such as an undelete program, tests for bad disk sectors, a copy utility which ignores CRC read errors, utilities for recovering files from a disk with a bad directory, and a single sector read/write/modify routine. The manual is very comprehensive and goes beyond merely describing how to use the utilities.

The package is available on either 5 or 8 inch disks and sells for only $75.00. Be sure to specify disk size and the cpu type (6800 or 6809).

## 9) SS-50 Computing

There is another magazine being published for the 6800/6809 world. It is called SS-50 Computing and comes to us from Utah. The magazine is in a small format, about 7 by 10 inches, but is completely typeset and very nicely done. It contains about 34 pages of 6800 and 6809 related reviews, editorials, press releases, programs, etc. It is published bi-monthly and can be subscribed to for $12.00 per year (6 issues) from:

SS-50 Computing
666 North Main
Logan, UT 84321

## 10) FLEX Based Products from Other Organizations

From time to time we like to let you know of good FLEX products we have had a chance to try out. We've got two this time...

a) DYNAMITE™ Disassembler

Computer Systems Center of St. Louis has long been a great supporter of the 68XX family and related products. They are now marketing a product of their own, a 6809/6800 disassembler called "DYNAMITE™" (DYNAMITE is a trademark of Computer Systems Center). This is a very powerful, very complete disassembler program that runs under 6809 FLEX. It can disassembler either 6809 or 6800 binary disk files. Output source can be routed to terminal, printer, or disk. It is a three pass disassembler and generates labels such that the resulting source can be reassembled as is or edited if desired. It is possible to define all data areas. A couple of unique features are the ability to specify a "command file" so that you don't have to reenter all the parameters each time you disassemble the same file and the ability to automatically use standard label names for specified addresses in the program. An example of the latter is that DYNAMITE can insert the correct label names for all the standard FLEX entry points in the disassembled program automatically. DYNAMITE costs $60.00 and is available from:

Computer Systems Center
13461 Olive Boulevard
Chesterfield, MO 63017

b) STYLOGRAPH™

Sonex Systems is offering a 6809 FLEX based word processing system called "STYLOGRAPH™" (STYLOGRAPH is a trademark of Sonex Systems). It boasts a cursor based or screen oriented editor and is available for several terminal types. Besides the editing capabilities, it also performs full formatting of text. STYLOGRAPH can take advantage of the many features of Qume, Diablo, and Spinwriter type printers to perform such functions as underlining, superscript, subscript, boldface, and proportional spacing. This is a very well thought-out and designed package. It is very powerful and yet not difficult to master. STYLOGRAPH sells for $150.00 for the specialty printers or $135.00 for a standard tty type printer from:

Sonex Systems
Box 238
Williamsville, NY 14221

## 11) A Couple of Quick FLEX Tips

Here are a couple of quick tips to use in your FLEX related software that you may not have thought of.

a) QUICK FILE TEST
Ever want to test a file to see if it could be read without errors? If you have our FLEX Diagnostics package there is a utility for doing just that. There is a crude method of performing such a test using the standard FLEX "LIST" utility. By placing a line number on the LIST command after the filename, it is possible to suppress all listing until that line is hit. If we specify a very large number such as 9999, the LIST command will never hit that line, but will have read through the entire file looking for it. Thus we have tested the file. This technique can also be applied to binary type files since they probably won't have many carriage returns in them which is what LIST looks for as line terminators. For example, to test the COPY command for read errors enter the command:

```
+++LIST 0.COPY.CMD 9999
```

b) DISABLE TWO SPACES AFTER PERIODS IN TEXT PROCESSOR
Our Text Processing System program always attempts to ensure that there are two spaces after every sentence punctuation. In other words, if the processor sees a period, question mark, or exclamation point followed by a space, it assumes it is at the end of a sentence and that there should be two spaces following the punctuation character. If that is not the case in the input text, the processor will insert a space in the formatted output. There are times when this is not desired. For example, consider the name "Mr. Jones". When the text processor saw the period followed by a space it would think it was at the end of a sentence and add a space. This would probably not be desired. A simple way around the situation is to be sure the period does not look like a sentence punctuation character. This is most easily accomplished by making the space that follows the period a non-paddable space character by use of the backslash. For example: "Mr.® Jones". Now the period is not followed by a real space, and it will not look like punctuation to the text processor.

## 12) Free FLEX Utility!

We are giving you another free FLEX utility in this newsletter. It dumps a binary file in the Motorola S1/S9 ASCII hex format and is called "S1DUMP". As written, the output is routed to the terminal. If desired, the user can route the output to a printer with the "P" command or to a disk file with the "O" command. Instructions for use are given in the listing itself. This listing is in 6800 code, but may be easily converted to 6809 by changing the equates and the ORG statement.

```
                    * S1DUMP UTILITY FOR 6800
                    *
                    * COPYRIGHT (C) 1980 BY
                    * TECHNICAL SYSTEMS CONSULTANTS, INC.
                    * PO BOX 2570, WEST LAFAYETTE, IN 47906
                    *
                    * THIS UTILITY DUMPS A BINARY FILE IN THE MOTOROLA
                    * S1/S9 ASCII HEX FORMAT.  OUTPUT IS SENT TO TERMINAL
                    * BUT CAN BE ROUTED TO DISK VIA THE FLEX "O" COMMAND.
                    * A SINGLE LINE OF TITLE CAN BE SENT WITH THE DUMP BY
                    * PLACING IT ON THE COMMAND LINE AS SHOWN HERE:
                    *     +++S1DUMP FILENAME THIS IS A TITLE
                    * WHERE "THIS IS A TITLE" WOULD BE OUTPUT BEFORE THE
                    * DATA AS A TITLE.  THE FILENAME DEFAULTS TO ".BIN".
                    *

                    * EQUATES

AD03                WARMS   EQU     $AD03
A840                FCB     EQU     $A840
AD18                PUTCHR  EQU     $AD18
AD24                PCRLF   EQU     $AD24
AD2D                GETFIL  EQU     $AD2D
AD33                SETEXT  EQU     $AD33
AD3F                RPTERR  EQU     $AD3F
AC14                BUFPNT  EQU     $AC14
B403                FMSCLS  EQU     $B403
B406                FMS     EQU     $B406

                    * MAIN PROGRAM

A100                        ORG     $A100

A100 20 08          S1DUMP  BRA     SDMP1
A102 01             VN      FCB     1           VERSION NUMBER 1

                    * TAPE OUTPUT CHARACTER ROUTINE (CAN BE PATCHED)
A103 7E AD 18       TOUCH   JMP     PUTCHR

                    * TEMPORARY STORAGE
A106                LENGTH  RMB     1
A107                ADDR    RMB     2
A109                CHKSUM  RMB     1

A10A CE A8 40       SDMP1   LDX     #FCB
A10D BD AD 2D               JSR     GETFIL      GET FILENAME FROM COMMAND
A110 24 03                  BCC     SDMP2
A112 7E A1 F9               JMP     ERROR
A115 4F             SDMP2   CLR A               DEFAULT EXTENSION IS .BIN
A116 BD AD 33               JSR     SETEXT
A119 CE A8 40               LDX     #FCB
A11C 86 01                  LDA A   #1          OPEN FOR READ
A11E A7 00                  STA A   0,X
```

```
A120 BD B4 06          JSR     FMS
A123 27 03             BEQ     SDMP3
A125 7E A1 F9          JMP     ERROR
A128 86 FF     SDMP3   LDA A   #$FF
A12A A7 3B             STA A   59,X        TURN OFF SPACE COMPRESSION
A12C BD AD 24          JSR     PCRLF
A12F FE AC 14          LDX     BUFPNT      POINT TO TITLE IN BUFFER
A132 BD A1 ED          JSR     PDATA2      SEND IT TO TAPE
A135 BD AD 24          JSR     PCRLF
A138 BD A1 C5  SDMP4   JSR     READ        GET A CHARACTER
A13B 81 16             CMP A   #$16        A TRANSFER ADDRESS?
A13D 26 08             BNE     SDMP5
A13F BD A1 C5          JSR     READ        IF SO, EAT UP ADDRESS
A142 BD A1 C5          JSR     READ
A145 20 F1             BRA     SDMP4
A147 81 02     SDMP5   CMP A   #$02        A BINARY RECORD HEADER?
A149 26 ED             BNE     SDMP4       IGNORE IF NOT
A14B BD A1 C5          JSR     READ        GET LOAD ADDRESS
A14E B7 A1 07          STA A   ADDR
A151 BD A1 C5          JSR     READ
A154 B7 A1 08          STA A   ADDR+1
A157 BD A1 C5          JSR     READ        GET RECORD LENGTH
A15A B7 A1 06          STA A   LENGTH
A15D B6 A1 06  SDMP6   LDA A   LENGTH
A160 27 D6             BEQ     SDMP4       NEXT RECORD IF FINISHED
A162 16                TAB
A163 C1 10             CMP B   #16         IS LENGTH > 16?
A165 23 02             BLS     SDMP7       SKIP IF NOT
A167 C6 10             LDA B   #16         ELSE, LIMIT TO 16
A169 10        SDMP7   SBA
A16A B7 A1 06          STA A   LENGTH      SET NEW LENGTH
A16D 8D 02             BSR     PUNREC      PUNCH ONE RECORD
A16F 20 EC             BRA     SDMP6       GO FINISH RECORD

                * ROUTINE TO PUNCH ONE RECORD WITH "B" EQUAL TO NUMBER
                * OF DATA BYTES TO BE PUNCHED.

A171 BD AD 24  PUNREC  JSR     PCRLF
A174 CE A2 02          LDX     #S1         PRINT HEADER
A177 BD A1 E0          JSR     PDATA
A17A 7F A1 09          CLR     CHKSUM
A17D 17                TBA
A17E 8B 03             ADD A   #3
A180 8D 20             BSR     OUTHEX      OUTPUT RECORD LENGTH
A182 B6 A1 07          LDA A   ADDR
A185 8D 1B             BSR     OUTHEX      OUTPUT LOAD ADDRESS
A187 B6 A1 08          LDA A   ADDR+1
A18A 8D 16             BSR     OUTHEX

A18C BD A1 C5  PUNRC1  JSR     READ        GET A DATA BYTE
A18F 8D 11             BSR     OUTHEX      OUTPUT IT
A191 FE A1 07          LDX     ADDR
A194 08                INX                 BUMP ADDRESS POINTER
```

```
A195 FF A1 07          STX     ADDR
A198 5A                DEC B
A199 26 F1             BNE     PUNRC1      LOOP IF NOT

A19B B6 A1 09          LDA A   CHKSUM      OUTPUT CHECKSUM
A19E 43                COM A
A19F 8D 01             BSR     OUTHEX
A1A1 39                RTS

                * OUTPUT BYTE IN "A" AS TWO ASCII HEX DIGITS

A1A2 36        OUTHEX  PSH A
A1A3 44                LSR A
A1A4 44                LSR A
A1A5 44                LSR A
A1A6 44                LSR A
A1A7 8B 90             ADD A   #$90
A1A9 19                DAA
A1AA 89 40             ADC A   #$40
A1AC 19                DAA
A1AD BD A1 03          JSR     TOUCH
A1B0 32                PUL A
A1B1 36                PSH A
A1B2 84 0F             AND A   #$0F
A1B4 8B 90             ADD A   #$90
A1B6 19                DAA
A1B7 89 40             ADC A   #$40
A1B9 19                DAA
A1BA BD A1 03          JSR     TOUCH
A1BD 32                PUL A
A1BE BB A1 09          ADD A   CHKSUM
A1C1 B7 A1 09          STA A   CHKSUM
A1C4 39                RTS

                * READ ONE CHARACTER FROM FILE

A1C5 CE A8 40  READ    LDX     #FCB
A1C8 BD B4 06          JSR     FMS
A1CB 26 01             BNE     READ1
A1CD 39                RTS                 RETURN IF GOOD
A1CE 32        READ1   PUL A               FIX STACK
A1CF 32                PUL A
A1D0 A6 01             LDA A   1,X         GET ERROR NUMBER
A1D2 81 08             CMP A   #8          END OF FILE ERROR?
A1D4 26 23             BNE     ERROR
A1D6 BD AD 24          JSR     PCRLF
A1D9 CE A2 05          LDX     #S9         PRINT END OF FILE MARKER
A1DC 8D 02             BSR     PDATA
A1DE 20 1C             BRA     EXIT        BACK TO FLEX

                * PRINT STRING TO TAPE

A1E0 A6 00     PDATA   LDA A   0,X
A1E2 81 04             CMP A   #4
```

```
A1E4 27 06            BEQ     PDATA1
A1E6 BD A1 03         JSR     TOUCH
A1E9 08               INX
A1EA 20 F4            BRA     PDATA
A1EC 39       PDATA1  RTS

A1ED A6 00    PDATA2  LDA A   0,X
A1EF 81 0D            CMP A   #$0D
A1F1 27 F9            BEQ     PDATA1
A1F3 BD A1 03         JSR     TOUCH
A1F6 08               INX
A1F7 20 F4            BRA     PDATA2

              * ERROR ROUTINE

A1F9 BD AD 3F ERROR   JSR     RPTERR
A1FC BD B4 03 EXIT    JSR     FMSCLS
A1FF 7E AD 03         JMP     WARMS

              * PRINT STRINGS

A202 53       S1      FCB     'S,'1,4
A205 53       S9      FCB     'S,'9,4


                      END     S1DUMP
```

NO ERROR(S) DETECTED