tiny c associates
post office box 269
holmdel, new jersey 07733

tiny-c
Newsletter

DATE: November, 1978
NUMBER: 1

1.0 Introduction

Tiny-c Associates will, from time to time, publish a newsletter for all registered owners of the tiny-c Owner's Manual. This is the first one. The intent is to share software ideas, bug notices and fixes, programming techniques, requests for new features and the like among tiny-c aficionados. Your participation in the Newsletter is welcome. Send your contributions to Tiny-c Associates, P. O. Box 269, Holmdel, New Jersey 07733. Be sure to indicate your contribution is for publication; if no clear indication is given, we will assume your letter is a private correspondence.

2.0 tiny-c/11 Addenda

Version 11-01-01 of tiny-c/11 appears in Appendix B of the tiny-c Owner's Manual. The fixes discussed in Sections 2.1 through 2.4 have already been incorporated into the machine-readable versions of tiny-c and apply only to this published version. However, the fixes described in Sections 2.5 and 2.6 should also be applied to the machine-readable Version 11-01-02.

2.1 Interpreter Subroutine REM (tiny-c/11-01-01)

Version 11-01-01 of REM did not begin by skipping blanks. This causes incorrect handling of comments which appear between a condition and its statement. Thus, the valid tiny-c construction

```
if(k>0)    /* a comment
   pn k
```

would cause an interpreter error. More satisfactory REM code is as follows:

```
REM:    MOV   CURSOR,R1          ; save CURSOR for diagnostic
        CMPB  #' ,@CURSOR        ; skip leading blanks
        BNE   1$
        CMP   CURSOR,PROGEND     ; check for run away
        BHIS  4$
        INC   CURSOR
        BR    REM
1$:     CMPB  #12,@CURSOR        ; jump over newlines
        BNE   2$
        INC   CURSOR
        BR    REM                ; a newline is a new game
2$:     MOV   CURSOR,R0
        CMPB  #'/,(R0)           ; check for a comment
        BNE   5$
        CMPB  #'*,1(R0)
        BNE   5$
        ADD   #2,CURSOR
3$:     MOV   CURSOR,R0          ; move to end of line
        INC   CURSOR
        CMPB  #12,(R0)
        BEQ   REM                ; a newline is a new game
        CMP   PROGEND,CURSOR     ; check for run away
        BHIS  3$
4$:     MOV   R1,CURSOR          ; restore CURSOR for diagnostic
        MOV   #CURSERR,-(SP)
        JSR   PC,@#ESET          ; report a cursor error
        TST   (SP)+
5$:     RTS   PC
```

2.2 Interpreter Subroutine SETARG (tiny-c/11-01-01)

The passing of a character variable to a function which declares the corresponding argument as an integer produced incorrect results in Version 11-01-01. This was due to SETARG's setting the two-byte integer value to the character byte and its adjacent byte rather than producing the second byte by extending the sign of the character byte. Better SETARG code is as follows:

```
SETARG: JSR   R5,SAVE
        MOV   4(R5),R0          ; set the argument's type
        CMPB  #'A,1(R0)
        BNE   1$
        MOV   4(R0),R2          ; if Actual, value is stuff
        BR    2$
1$:     MOV   4(R0),R3          ; else what stuff points to
        MOVB  (R3),(SP)
        MOVB  1(R3),1(SP)
        MOV   (SP),R2
2$:     TSTB  (R0)              ; test for a one-byter
        BNE   3$
        CMPB  #1,2(R0)
        BNE   3$
        MOVB  R2,R2             ; extend the sign
3$:     MOV   R2,(SP)           ; allocate the value
        MOVB  6(R5),R0
        MOV   R0,-(SP)
        JSR   PC,@#VALLOC
        JMP   RETURN
```

2.3 The Call to USERMC in MC (tiny-c/11-01-01)

The code which sets up the machine stack for a call to USERMC in Version 11-01-01 of MC is incorrect. The correct code is as follows:

```
MOV   4(R5),(SP)        ; load number of arguments
MOV   -10(R5),-(SP)     ; load function number
SUB   #1000.,(SP)       ; subtract 1000 from it
JSR   PC,@#USERMC       ; call USERMC
```

2.4 The In Array in FPS (tiny-c/11-01-01)

To accomodate the longer terminal input lines typical of DEC installations, the input array, ln or line, in FPS should be dimensioned at 133 rather than 64.

2.5 Channel Number as Table Offset in TCIO (tiny-c/11-01-02)

There are five places in TCIO (once in OPEN, and twice each in GETBL and PUTBL) where the channel number in R3 is used to address the BLOCK table as BLOCK(R3). Since BLOCK is a word-entry table, this causes a system trap when odd-numbered channels are being accessed. The easiest fix is to do an ASL R3 before the BLOCK(R3) instructions and an ASR R3 after the instructions. Alternatively, the five instructions using BLOCK(R3) could be changed to byte instructions. We thank John Osudar of Homewood, Illinois for pointing this out.

2.6 Transient Value Changes in Interrupt Systems (tiny-c/11-01-02)

Unfortunately, the tiny-c/11 subroutine TCTOI which returns a full word from two adjacent byte addresses in memory uses the word below the word the stack pointer is pointing to form the full word. Should an interrupt occur after the word has been formed here but before it is moved to R0, the value that is actually returned is the PSW at the time of the interrupt. Obviously, this can produce some strange results and be quite maddeningly unreproducable. It is HIGHLY RECOMMENDED that the following patch be applied to all tiny-c/11-01-02 systems:

```
.RUN PATCH

FILE NAME--
*SY:TINYC.SAV
*11720/ 116666  116646  <LF>
11722/  4  <LF>
11724/  177776  116666  <LF>
11726/  116666  000004  <LF>
11730/  2       000001  <LF>
11732/  177777  012600  <LF>
11734/  16600   000207  <LF>
11736/  177776  000240  <LF>
11740/  207     000240  <CR>
*E
```

This patch corresponds to the following new version of TCTOI:

```
TCTOI:  MOVB    4(SP),-(SP)
        MOVB    4(SP),1(SP)
        MOV     (SP)+,R0
        RTS     PC
        NOP
        NOP
        .END
```

2.7 tiny-c/11 Running under RT-11/FB (tiny-c/11-01-02)

Dr. Charles Retter of Framingham, Massachusetts notes that bit 6 in the JSW must be set if the .TTINR request in TSTCHR is to perform satisfactorily. If the bit is not set, the RT-11 monitor will block the execution of the Job until a character is received, defeating the purpose of TSTCHR. Dr. Retter also notes that odd buffer addresses to .WRITW under FB will cause an odd address trap. We have not observed this problem with RT-11 V03B, and would welcome suggestions for dealing with it from other RT-11/FB users.

3.0 tiny-c/8080 Addenda

3.1 Corrected Listings (tiny-c/8080-01-01)

Three instructions on the first page of the Appendix A 8080 listings have wrong addresses. These are:

wrong version		correction	
address		address	
2000	C3 58 26	2000	C3 58 2C
2003	C3 7E 26	2003	C3 7E 2C
2006	C3 81 26	2006	C3 81 2C

The machine-readable media are correct.

3.2 Bug Fixes (tiny-c/8080-01-01)

The contents of address 2D27 should be C4 (CNZ). The listings in Appendix A show CC (CZ) and the media may have CD (CALL).

At location 30E8, an LXI B,1 is needed before the CALL FCLOSE for some operating systems.

In Chapter VI, page 6-12 (second page of Figure 6-1) the lines at 6044 should read:

```
6044  F8  30    0400  DW  30F8H
```

In line 604C change the D5 to D5.

3.3 FREAD Definition Clarified (tiny-c/8080)

David McWherter, Jr., of Cornwells Heights, Pennsylvania, points out that FREAD shouldn't return an EOF unless the returned count is zero. The EOF signal is to be returned on the FREAD call after the last record of data is returned. It does not accompany the last record.

3.4 Relocater Bug

The relocater program in Figure 6-1, page 6-11, of the tiny-c Owner's Manual does not properly relocate the address at 2D3A. This address will have to be relocated manually and should be set to -BUFF-1.

4.0 PPS Errata and Addenda

4.1 Corrections to Appendix C--Crunched PPS Code

The following corrections should be made to the "crunched" version of PPS which appears in Appendix C of the tiny-c Owner's Manual. The version of PPS in Chapter IV and those distributed on machine-readable media are correct.

page	function	line	from	to
5	main	5	pl(0)=13	pr(0)=13
6	fc	4	k=k+1	k=k-1
7	oi	13	no()!=0	nl()!=0
8	ch	7	((	(
9	up	4	else(	else{

Thanks to Dale Walker of New York City for the second correction, and again to David McWherter for the other four.

4.2 Functions Omitted from Chapter IV PPS

The version of PPS in Chapter IV does not include the functions chrdy, fopen, fread, fwrite, or fclose. Appendix C does include these functions. The machine-readable media also include these functions, and may, in fact, include still others as noted in the appropriate installer's guides.

4.3 index Documentation

In the documentation for the library function index on page 2-24 of the tiny-c Owner's Manual, the words "OF" and "IN" should be interchanged.

4.4 Line 0 Errors

On page 3-2 of the tiny-c Owner's Manual, line 0 of the program buffer in PPS is defined to be just a carriage return. Nevertheless, PPS can produce error messages which reference line 0, especially:

```
0 -- err 26

<
```

These error messages actually apply to the console input line and not to the program in the program buffer. Error 26 usually means that you tried to initiate the execution of a function which could not be found in the program buffer. Thanks once more to John Osudar for pointing this out.

tiny-c
Newsletter

DATE: May, 1979
NUMBER: 2

1.0 New Products

The tiny-c interpreter and Program Preparation System are available in six new formats: TRS-80 cassette; CP/M 8" soft-sectored and Micropolis 5" dual or quad density diskette; North Star DOS 5" single density diskette; and a PDP-11 to 8080 cross-assembled version.

The TRS-80 cassette version is recorded in Level II SYSTEM format and includes line printer and graphics support. It also reads and writes EDTASM compatible files. At least 16K bytes of random access memory are recommended for its effective use. The load-and-go TRS-80 cassette of tiny-c costs $30.

The CP/M and North Star installations are fully interfaced to their respective disk operating systems. The CP/M diskette also contains the source code of the interpreter. The North Star version loads at 2A00. Both the CP/M and North Star diskette versions of tiny-c are $35.

The PDP-11 to 8080 cross-assembled version of tiny-c consists of the 8080 source code of the tiny-c interpreter on a PDP-11 diskette together with a full set of macros to assemble the 8080 code with the DEC MACRO-11 assembler. The PDP-11/8080 cross-assembed diskette costs $35.

tiny-c continues to be available for the SOL-20 on Helios diskette and CUTS cassette, on Poly-88 and Tarbell cassettes, and on DEC RT-11 diskette.

2.0 Full C Compiler

A full C compiler for most DEC operating systems is now available through tiny c associates. The compiler, by Whitesmiths, Ltd., handles bit fields and casts and comes with an extensive runtime library. Binary licenses are $500 and source licenses $5000. Contact Scott Guthery at (609) 443-3992 for details.

3.0 Computer Shows

tiny c will have booths at the Trenton Computer Festival, the West Coast Computer Faire, and the Philadelphia Computer Show this year. Be sure to stop by and say "Hi".

4.0 Program Preparation System Fixes

Here are some bug fixes and improvements for PPS:

4.1 Improved backward scan

The following change to PPS makes the -n command work with blinding speed. On page 9 of Appendix C, replace the up function with:

```
up [
int v(1)
if(ln(1)==0) v(0)=1
else num(ln+1,v)
if((v(0)=lo-v(0))<0) v(0)=0
so(v)
]
```

4.2 Problems with too many DELs

There is a bug in the function ss on page 1 of Appendix C which causes problems if too many DELs are typed. The fifth line needs a semicolon so that it reads:

```
else if(b(1)==127)[if(l>0)l=l-1;]
```

This fix is required on all media labeled 80-01-01. Note that the constant 127 is installation dependent so if you find a different constant in your PPS, don't change it.

4.3 Typo in in function

In the in function in Appendix C on page 8, line 31, ll==ll+1 should read

```
ll=ll+1
```

The machine readable media are correct. Thanks to Robert Pasky of Waltham, Massachusetts, for this one.

4.4 Error return without a close

In the readfile function in Appendix C on page 4, line 9, the "if(k==-1) return t" statement should read

```
if(k==-1){MC(u,6);return t;}
```

so that the unit on which the readfile error was raised is closed before returning. This fix is required on some, but not all, media labeled 80-01-01, and some may already have it. Even though some installations don't need to worry about this it's safer to have it even if it's not required. Thanks to Ray Duncan of La Verne, California, for calling this one to our attention.

4.5 Further Problems with fc

In the fc function in Appendix C on page 6, line 14, the while statement should read:

```
while(pr(k=k-1)!=13)if(k<=0)break
```

which stops an occassional PPS abort with a subscript out of range error.

4.6 Null locate

If the PPS command line

```
.1/<CR>
```

is entered, a subscript underflow error is raised due to the check for the caret character at the end of the string to be located. This can be prevented by placing the statement

```
if(ln(3)==0)return
```

as the first executed statement in the function oi in Appendix C on page 7.

4.7 PPS as an application program

A few tiny-c users called in with this problem:

> "I was using PPS to edit PPS. The I started the edited PPS to test it. So far, so good. I used this PPS (running as an application) to prepare a small program. Still OK. I even ran the program (as a second level application), and stopped it, edited it, and ran it again. Everything was just fine. Then I aborted the application level PPS to return to the system level one. And the system crashed!!!"

The problem is the endlibrary statement, which is permitted only at the system level. If you want to TEST a PPS at the application level, remove its endlibrary statement. Then test it. Then put the endlibrary statement back before recording it.

Some other notes. To run a PPS at application level, be sure to reduce the pr dimension and, 4 lines later, the lp assignment value. Two or three hundred bytes is enough space in pr to test a new PPS. Remember you have two copies of PPS in RAM and two complete sets of its variables.

If you change the length of the input line ln in its dimensions statement be sure to put the new length in the while condition in the st function.

By the way, before using PPS to edit PPS make sure the original dimension of the pr array is large enough to accomodate PPS itself. PPS is about 5015 bytes so an original dimension of 5200 or more should suffice. Some versions of PPS are delivered with a dimension of 5000 and these will have to be changed using the standard system edit. Remember to change both the dimension of pr and the assignment four lines later.

4.8 Null filename reads and writes

There is currently no check in the readfile or writefile routines for a null (zero length) filename. Some operating systems tolerate this condition, some do not. If you have any problems with a .r or .w command with no filename, you should put a check for a null filename in readfile and writefile.

4.9 PPS extensions

Don de Courcelle of Avenel, New Jersey, sends along the following helpful extensions for PPS. Many thanks to Don and kudos to his emphasis on user oriented computing.

FOR THOSE USERS THAT ARE TIRED OF REFERING BACK TO THE ERROR MESSAGE TABLE IN THE BACK OF CHAPTER 3, I RECOMMEND THESE 2 ROUTINES REPLACE THOSE FOUND IN THE STANDARD PPS (ON PAGES 9 AND 10 OF APPENDIX C). THEY ALSO IMPLIMENT ERROR HANDLING FOR IMMEDIATE LINE-INPUT EXECUTION (FORMERLY REPORTED AS LINE 0 ERRORS WITH NO DISPLAYED TEXT)...

```
WH
[INT F, L, U, B, X
 PL "#EXEC FAULT @ LINE"
 PN LO; PS" -- ERR: ";
 X=ER(0)
 IF(X==1) PS"BAD STMT"
 ELSE IF(X==2) PS"CURSOR EOF"
 ELSE IF(X==3) PS"ID REQD"
 ELSE IF(X==5) PS"MISSING ')'"
 ELSE IF(X==6) PS"INV INDEX"
 ELSE IF(X==7) PS"BAD PTR EXPR"
 ELSE IF(X==9) PS"AEXP REQD"
 ELSE IF(X==14) PS"BAD ASSIGN"
 ELSE IF(X==16) PS"STK OVFLO"
 ELSE IF(X==17) PS"FUNC OVFLO"
 ELSE IF(X==18) PS"VARI OVFLO"
 ELSE IF(X==19) PS"OUT OF MEMORY"
 ELSE IF(X==20) PS"STARTUP ERR"
 ELSE IF(X==21) PS"WRONG NUM OF PARMS"
 ELSE IF(X==22) PS"FUNC BODY ERR"
 ELSE IF(X==24) PS"BAD MC"
 ELSE PS"UNDEF SYMBOL"
 PL""
 IF (CU < 0)
  [INT I
   WHILE( (LL=LL-1) > 0)  PUTCHAR( LN(I=I+1) );  PL""
   B=64+CU
   CU=0
  ]
 ELSE
  [U=CU
   F=FC
   B=U-F
   L=LC
   F=F-1
   WHILE( (F=F+1) < L)  PUTCHAR( PR(F) ); PL""
  ]
 WHILE( (B=B-1) >= 0) PUTCHAR(' ')
 PS "<--- ERR ---<<";  PL""
]


ST
[INT LLX, SVCU
 LLX=LL
 WHILE( LLX <= 64 )
  [LN(LLX)=' '
   LLX=LLX+1
  ]
 MC ER, LN+1, PR+PE, PR, 11
 IF(CU < 0)
  [LN(LL)=0 /* RESTORE END OF LINE NULL
   SVCU=CU
   CU=0
  ]
 ELSE
 IF(CU > PE) CU=PE
```

```
 LO=COUNTCH(PR, PR+CU-1, 13)
 IF( ER(0) )
   IF( ER(0)==99)  [ PL"#ESC ABORTED"; PL""]
   ELSE
    [IF(SVCU != 0)  CU=SVCU
     WH
    ]
]
```

FOR THOSE USERS WHO HAVE A CRT TERMINAL, REPLACING THIS ROUTINE FOR THE ONE IN PPS CAN PROVE ENJOYABLE (REFER TO PAGE 1 - APPENDIX C). IT TURNS YOUR "DEL" KEY INTO A GENUINE BACKSPACE !!

```
GS CHAR B(0)
[INT I;  CHAR C
 I=-1
 WHILE (1)
  [C = B(I=I+1) = MC 2
   IF(C==13) [B(I)=0; RETURN I]
   ELSE
   IF(C==27) [PS" <<DEL<<"; PL""; I=-1]
   ELSE
   IF(C==127)
     IF(I >= 1)
      [PUTCHAR(8)
       PUTCHAR(' ')
       PUTCHAR(8)
       I=I-2
      ]
     ELSE I=I-1
    ]
]
```

FOR THOSE USERS THAT WOULD LIKE TO EDIT LARGER LINES OR FILES USED THESE CHANGES IN YOU PPS (REFER APPENDIX C):

```
>>> ON PAGE 5, DELETE LINE 3>>      CHAR FT(20),TT(20)
>>> CHANGE LINE 4 TO:               INT FL, TL, LNSZ, PRSZ
>>> CHANGE LINE 5 TO:               CHAR LN(LNSZ=80), PR(PRSZ=8000)
>>> INSERT AFTER LINE 5:            CHAR FT(LNSZ), TT(LNSZ)
>>> CHANGE LINE 9 TO:               LP=PRSZ
>>> IF YOU USE THE NEW "WH"         B=LNSZ+CU
    ROUTINE GIVEN ABV USE THIS
    THIS INSTEAD OF "B=64+CU".
>>> IF YOU USE THE NEW "ST"         WHILE( LLX <= LNSZ )
    ROUTINE GIVEN ABV USE THIS
    TO REPLACE "WHILE(LLX<=64)"
    OTHERWISE CHANGE "64" TO
    "LNSZ" IN LINE 2 OF PAGE 10.
```

FOR THOSE USERS DESIRING A MORE HUMANIZED ./ COMMAND SUBSTITUTE THIS ROUTINE INTO PPS (APPENDIX C, PAGE 9):

```
^A
[PS"CURRENT LINE:"; PN LO
 PS"  LINES:"; PN LA
 PS"  BYTES USED:"; PN PE
 PS"  BYTES LEFT:"; PN LP-PE; PL""
]
```

THIS CONVENIENCE CHANGE MAKES A SINGLE CARRIGE-RETURN INPUT AT THE BEGINING OF A LINE ACT EXACTLY LIKE A "+ CARRIGE-RETURN", INCREMENTING THE LINE POINTER AND DISPLAYING THE NEXT LINE. (REFER TO APPENDIX C).

```
>>> CHANGE LINE 13, PG 5:          LL=GS(LN)
>>> AFTER THIS LINE >>             ELSE IF(C=='+') DO
    INSERT THIS LINE:              ELSE IF(C==0) NEXT
>>> AFTER FUNCTION "DO"            NEXT [ LN(1)=0; DO ]
    ON PG 9 INSERT THIS
    NEW FUNCTION.
```

5.0 8080 Fixes

Here is a collection of fixes and improvements for the 8080 version of tiny-c.

5.1 Ordering

The POP D and JC PERR are in the wrong order at 21DE. Put the POP D before the JC PERR. This fix is required on all media labeled 80-01-01.

5.2 Typos

The last two listing lines in Appendix A should read

```
30F4    3A 93 20
30F7    C3 59 2D
```

The distributed media are correct. Note that there are three typos corrected here.

Another typo is noted in 5.4a below.

5.3 Misleading comment

In line 217C the comment incorrectly describes the action of DCMP, which does not set Z as described. Delete "Z," from the comment.

5.4 Assembly from Appendix A

For those enterins the 8080 source code from Appendix A and assemblins it, the followins chanses may be required dependins on the properties of your assembler:

a) At 2782, B7 ORA A,E should be B3 ORA A. Some media require correctins the object code byte.

b) At 2209, MVI 0,B should be MVI B,0

c) The labels D2, D3, D4, DONE, and WHERE are duplicated. Further, PUSH, POP, and OUT are used as labels. Make the followins chanses:

- in DREM (2131 to 216E) chanse all occurances of D2, D3, and D4 to DR2, DR3, and DR4 respectively

- in BLANKS (2243 to 2253) chanse all occurences of OUT to BLOUT

-in FACTOR (26FB to 282A) chanse all occurances of WHERE to FWHERE

-at 2CD8 and 2D00 change DONE to DONEMSG

-change the subroutine names PUSH (21CD) and POP (21A9) to PUSHST and POPST respectively, and reflect these changes in all CALLs to these routines.

Thanks to James A. Petroski for pointing these items out.

5.5 Using the <CR><LF> pair as end-of-line

The following patches make the 8080 interpreter allow either <CR> or the pair <CR><LF> as the end-of-line indicator:

Change REM as follows:

```
replace
        2349    JNZ     REM
        234C    LXI     D,CMNT

with
        JZ      RE2
  RE3   MOV A,M
        CPI     0AH ; <LF>
        JNZ     REM
        INX     H
        SHLD    CURSOR
        JMP     REM
  RE2   LXI     D,CMNT

and replace
        235B    JMP     REM

with
                JMP     RE3
```

Change SKIPST as follows:

```
replace
        286D    JZ      SS5

with            JZ      SS8

and replace
        2883    SHLD    CURSOR

with    SS8     SHLD    CURSOR
```

5.6 Stack overflows

In the 8080 version if any of the areas STACK, FUN, or VAR is exceeded, a few bytes beyond its Ending address are overwritten. (See Section 6.5.2) This can cause unexpected DONE messages. A small guard area at the end of each space will protect against this.

After calculating the eight allocation addresses, add 5 to ESTACK, 6 to EFUN, and E(hex) to EVAR. So the example on page 6-19 of the tiny-c Owner's Manual would read:

	begin	-end-1+guard
	=====	==========
STACK	1200	ED85
FUN	1280	ED06
VAR	1300	E60E
PR	1A00	C000

There is no way to guard against an overflowing of the 8080 machine stack. Be sure to allocate plenty of RAM for it. 1000 bytes should usually be enough. A highly recursive application may require more however.

6.0 CP/M Fixes

The following fixes should be applied to CP/M disks labeled 80-01-01. Disks labeled 80-01-01b have had these fixes already applied to them.

6.1 Incorrect default DMA address

In CTC.ASM, line 110 should read

```
        LXI     D,80H
```

Patch TC.COM as follows:

```
A> ddt tc.com
- l157
(erroneous TC.COM will show LXI D,0500)
- a157
157 LXI D,80
15A .
- l157
(corrected TC.COM will show LXI D,0080)
- <cntl>C
A> save 23 tc.com
```

The fifth line from the bottom of the CP/M version of PPS reads

```
k=81+32*(k%4)
```

Change the 81 to a 129.

6.2 Missing right parenthesis

The fourteenth line needs a right parenthesis after the 127. If it is not there, this was the cause of the <cntl>U blowups.

7.0 Programming Technique

The MC 11 library machine call can be used to overlay large application programs. The general scheme is as follows. Prepare a "root segment" file which looks like this:

```
library routines used by most overlays
endlibrary statement
super globals ... communication between overlays
main program that does overlay work
```

One of the super globals is a filename. The main program works as follows:

```
set filename to 1st overlay
while(filename is not finish signal) [
        read the named file
        use MC 11 to execute it
        ]
```

Overlays are written with the following protocol: to cause control to pass to another overlay, put that overlay's filename into the filename super global and return. To cause the whole program to stop, put the finish signal in the filename super global and return.

Data can be passed among the overlays using other superglobals.

An overlay cannot have an endlibrary statement in it (see 4.7 above).

If somebody comes up with a specific implementation or application of overlays we will pass it along in the next Newsletter.