EMPL

USER'S MANUAL

July 18,1977

Erik T. Mueller
Britton House
Roosevelt, NJ 08555
(609) 448-2605

LOADING 8MPL

1) Enter the following checksum loader into memory. (All locations are split octal.)

```
037  321      061        LXI SP
     322      320
     323      037
     324      016        MOV C,0
     325      000
     326      076        MVI A,020
     327      020
     330      323        OUT 156
     331      156
     332      041        LXI H,0
     333      000
     334      000
     335      315 LOOP   CAL INTP
     336      366
     337      037
     340      373        EI
     341      167        MOV M,A
     342      201        ADD C
     343      117        MOV C,A
     344      043        INX H
     345      174        MOV A,H
     346      376        CPI 026
     347      026
     350      302        JNZ LOOP
     351      335
     352      037
     353      315        CAL INTP
     354      366
     355      037
     356      271        CMP C
     357      302 CSER   JNZ CSER
     360      357
     361      037
     362      363        DI
     363      303 GOOD   JMP GOOD
     364      363
     365      037
     366      333 INTP   IN  156
     367      156
     370      346        ANI 020
     371      020
     372      302        JNZ INTP
     373      366
     374      037
     375      333        IN  157
     376      157
     377      311        RET
```

2) Put tape into cassette recorder.
3) Examine 037 321; Run
4) Start tape
5) When the INTE LED goes on, 8MPL is being read into memory.
6) When the address lights change to 037 367 and the INTE LED goes out, the read is complete. If the address lights instead indicate 037 377 with the INTE LED still on, there has been a checksum error and the user must try again.

Implementing 8MPL on your system

1) Store the address of the first byte available for the workspace.
   025 260 LLL (Low byte of address) — ØØØ
   025 261 HHH (High byte of address) — Ø52
   025 244 LLL — ØØØ
   025 245 HHH — Ø52
   (Since 8MPL resides from 000000 to 025377, the first available byte would normally be 026000 unless the user wishes to put I/O or other subroutines right after 8MPL.)

2) Store the address of the first byte available for the workspace + 2.
   000 105 LLL — ØØ2
   000 106 HHH — Ø52
   (This would normally be 026002.)

3) Store the address of the last byte available for the workspace.
   000 001 LLL — 377
   000 002 HHH — 137
   002 335 LLL — 377
   002 336 HHH — 137
   016 276 LLL — 377
   016 277 HHH — 137
   023 137 LLL — 377
   023 140 HHH — 137
   023 331 LLL — 377
   023 332 HHH — 137
   (In an 8K system, this would be 037 377 if I/O subroutines are right after 8MPL or in a monitor somewhere else in memory.)

4) Store the address of your INPUT subroutine.
   000 011 LLL — ØØØ
   000 012 HHH — Ø26
   Your INPUT subroutine should get a character from the terminal and put it in the Accumulator (A). Bit 7 should be 1. The character should not be echoed, as 8MPL will use your OUTPUT subroutine to echo it later. No registers other than A may be changed.
   INPUT resides at Ø26ØØØ.

5) Store your data input from keyboard port number. This is used to check for Control C. (Program interrupt)
   012 330 xxx — ØØ1

6) Store the address of your OUTPUT subroutine.
   015 072 LLL — 006
   015 073 HHH — 026
Your OUTPUT subroutine should print the character in A on your terminal. Bit 7 is 1. No registers including A may be changed.
When a CR (215) is output, your subroutine should also output a LF (if necessary on your terminal.) Remember that in this case, a CR must be left in A upon exiting the subroutine.
OUTPUT resides at 026006

7) Store the address of your subroutine to clear the screen (if applicable.)
   014 333 LLL — 045 @CALF instead
   014 334 HHH — 026
If you have no such subroutine, store 014 333 060.
   334 000

8) Store the character which will cause your terminal to backspace (or the character that you wish to be echoed when Control H is typed.)
This would be a Curser Left on a CRT terminal.
   015 260 xxx — 377
377 translated to 010 on output.

9) Store the character which will cause your terminal to forward space without changing the character on the screen. (This will be echoed when a Control F is typed.)
This would be a Curser Right on a CRT terminal, or simply a space (240) on a prigting terminal.
   015 302 xxx — 040 (space)

10) Store the address of your monitor or operating system. (Used for the)QUIT command)
   025 263 LLL — 050
   025 264 HHH — 040
11) Make a new tape of EMPL for future use.

EMPL is now ready to run!

RUNNING EMPL

1) To enter EMPL and create a clear workspace, start execution at 000 000.

2) To exit EMPL, type)QUIT.
To re-enter EMPL leaving the workspace intact, start execution at 000070.

3) To save a workspace on tape:
   a) Type)QUIT. (Quit must be used to exit EMPL, as it stores essential data in workspace for taping.)
   b) Dump memory onto tape starting at the location at 025 260 (L) 025 261 (H), and stopping at the location at 023 027 (L) 023 030 (H).

4) To load a workspace from tape:
   a) Load workspace back in memory from tape.
   b) Start execution at 000070 .

Note: If the workspace is being loaded into a different part of memory than it was originally in, follow the following directions.
   a) Load workspace back into memory from tape.
   b) Store the new end of tape address at the location at 025 260 (L) 025 261 (H).
   c) Start execution at 000070.

EMPL TAPE MAKER

Here is the program I used to make to MMPL tape:

```
037 000 061        LXI SP
    001 377
    002 037
    003 041        LXI H,0
    004 000
    005 000
    006 076        MVI A,074
    007 074
    010 315        CAL OTP
    011 046
    012 037
    013 076        MVI A,346
    014 346
    015 315        CAL OTP
    016 046
    017 037
    020 016        MVI C,0
    021 000
    022 176 LP,    MOV A,M
    023 315        CAL OTP
    024 046
    025 037
    026 201        ADD C
    027 117        MOV C,A
    030 043        INX H
    031 174        MOV A,H
    032 376        CPI 026
    033 026
    034 302        JNZ LP
    035 022
    036 037
    037 171        MOV A,C
    040 315        CAL OTP
    041 046
    042 037
    043 303 SELF,JMP SELF
    044 043
    045 037
    046 365 OTP,   PUSH PSW
    047 333 WT,    IN   156
    050 156
    051 346        ANI  040
    052 040
    053 302        JNZ WT
    054 047
    055 037
    056 361        POP PSW
    057 323        OUT 157
    060 157
    061 311        RET
```

Workspace Organization

```
Workspace:    Functions Variables
Function:     Header (l) (h)  (Prog  Monadic  Dyadic)
                     \_____/
               Add to address of H byte
              for address of next object
Prog:         Text
Monadic:      Result (377) Rightvar (377) Text
Dyadic:       Result (377) Leftvar (377) Rightvar (377) Text
Variable:     Header (#) (l) (h) [(L) (H)]#L
              String or  # of      element HL
               numeric  elements   of vector
Header:       Name (377) (K00)
Text:         (l)(h)  (#) [Line  (or)]∞0 (377)(377)
              line   bytes
             number  in line

         If they meet, WS FULL error.

State indicator:   (377) (377) Return address Stacked data
Return address:    PRM: Program called from another program
                   PMR: Program was used as a function
                   REX: Return to execution mode
Stacked data:      Expression data  State indicator


K00 = 010 Program
      011 Dyadic function
      012 Monadic function
      004 Variable
```

```
MPLIO 2A00  3210


1600                    0010 ********************************************
1600                    0020 *          I/O FOR EMPL/N.S. DOS INTERFACE    *
1600                    0030 *          STEVE WILLIAMS 8/18/?? BYPA        *
1600                    0040 ********************************************
1600                    0050 *       IN ADDITION TO THIS APPENDATION TO ERIK M
UELLER'S
1600                    0060 *       EMPL, SEVERAL PATCHES HAVE BEEN MADE TO T
HE EMPL
1600                    0070 *       ITSELF. THESE ARE DOCUMENTED IN THE EMPL
MANUAL
1600                    0080 *       SUPPLIED WITH YOUR COPY. THIS I/O PACKAGE
, AND THE
1600                    0090 *       PATCHES MENTIONED CONSTITUTE THE STANDARD
 BYTE OF
1600                    0100 *       PALO ALTO EMPL/NORTH STAR CONFIGURATION,
AND ARE
1600                    0110 *       INTENDED FOR THE SOLE USE OF THE CUSTOMER
S WE SUPPLY.
1600                    0120         ORG   1600H      ;LIVES AT END OF EMPL.
1600                    0125 PSW     EQU   6          ;KLUDGE FOR SCS.
1600                    0130 DCHIN   EQU   2010H      ;DOS INPUT CHARACTER.
1600                    0140 DCHOUT  EQU   200DH      ;DOS OUTPUT CHARACTER.
1600                    0150 DOS     EQU   2003H      ;DOS RESTART.
1600                    0160 CR      EQU   0DH
1600                    0170 LF      EQU   0AH
1600                    0175 BS      EQU   8
1600                    0180 *
1600                    0190 *       INPUT CHARACTER.
1600                    0200 *
1600 CD 10 20           0210 CHIN    CALL  DCHIN      ;GET A CHARACTER FROM DOS.
1603 F6 80              0220         ORI   80H        ;SET HIGH BIT.
1605 C9                 0230         RET              ;RETURN.
1606                    0240 *
1606                    0250 *       OUTPUT A CHAR
1606                    0260 *
1606 F5                 0270 CHOUT   PUSH  PSW        ;SAVE A
1607 C5                 0280         PUSH  B          ;SAVE B
1608 E6 7F              0290         ANI   7FH        ;STRIP TOP BIT
160A FE 0D              0300         CPI   CR         ;CHAR A CR?
160C CC 1D 16           0310         CZ    LFEED      ;   YES:DO A LINEFEED FIRST.

160F FE 7F              0312         CPI   7FH        ;USE THIS AS A BACKSPACE.
1611 C2 16 16           0314         JNZ   NOTBS      ;TRANSLATE.
1614 3E 08              0316         MVI   A,BS
1616 47                 0320 NOTBS   MOV   B,A        ;CHAR TO B FOR DOS.
1617 CD 0D 20           0330         CALL  DCHOUT     ;DO OUTPUT.
161A C1                 0340         POP   B          ;RESTORE REGS.
161B F1                 0350         POP   PSW
161C C9                 0360         RET              ;RETURN
161D 3E 0A              0370 LFEED   MVI   A,LF       ;GET A LINEFEED.
161F CD 06 16           0380         CALL  CHOUT      ;PRINT IT.
1622 3E 0D              0390         MVI   A,CR       ;GET CR BACK.
1624 C9                 0400         RET
1625                    0410 *
1625                    0420 *       AT-SIGN ,CRLF. INSTEAD OF CLRSCR.
1625                    0430 *
1625 3E 40              0440 CLR$CR  MVI   A,40H      ;AT-SIGN.
1627 CD 06 16           0450         CALL  CHOUT
162A 3E 0D              0460         MVI   A,CR       ;CRLF.
162C C3 06 16           0470         JMP   CHOUT      ;PRINT AND RETURN FROM THER
E.
162F                    9998 *       END
```

E M P L , an 8080 APL

Britton House
Roosevelt, N.J. 08555

8K EMPL is a micro APL for the Intel 8080. The current version fits in the first 5,632 bytes of memory. All the special symbols and operators of APL have been adapted to the ASCII character set.

The following description explains EMPL. Examples are used to clarify all operations.

EMPL has two modes: The normal Execution Mode in which all instructions are executed immediately, and the Definition Mode which permits the user to enter functions.
When in Execution Mode, the computer indents five spaces and waits for input.
Type an arithmetic expression and EMPL will respond with the result.

```
      2-6-3
-1
```

(Computer responses are underlined.)
All expressions are evaluated by the right-to-left rule. That is, each operator takes as its right argument everything to its right. For example, in the above problem, EMPL first evaluates 6-3 which equals 3, then it evaluates 2-3 which equals -1.
The range is ±32767-- double-byte integer arithmetic is used.
Variable names may be any length of alphabetic characters. They are assigned values with the ":=" (typed as Control I.)

```
      SPEED:=2059
      SPEED - 59
2000
```

One dimensional arrays are called vectors in EMPL. A variable may be specified to be a vector of any length provided sufficient memory is available.

```
      A:= 5 -4 2 33
      A
5 -4 2 33
      A+1
6 -3 3 34
      A*4 3 2 0
20 -12 8 0
```

A Dyadic Operator (such as "+","*","/",etc.) takes two arguments, one on either side of the operator; as compared with a Monadic Operator such as "!" (absolute value), or "?" (random), which takes only one argument to its right.
Constants may be numeric or string vectors. Since the internal representation of both is the same, operations may be performed between the two types. To print a vector as a string, type "$" before it.

```
      X:='STRING'
      X
83 84 82 73 78 71
      $X
STRING
      $X+1
TUSJOH
```


A description of some of EMPL's unique operators follows:
"\" is used to create a vector of consecutive integers between 1 and the right argument.

```
      \5
1 2 3 4 5
```

"^" gives the length of the right argument.

```
      ^'HELLO'
5
      ^\23
23
```

"!" returns the smaller of two numbers; """ returns the greater.

```
      -1'2
-1
      4 1 9"3 2 56
4 2 56
```

"|" gives the remainder when the right argument is divided by the left.

```
      2.5
1
```

"<","<="(Control Y),">",">="(Control N),"=", and "#" return a 1 if the condition is true, otherwise they return a 0.

```
      5<4
0
      7 2 4#\3
1 0 1
```

"," chains together two vectors.

```
      HI:= 1 2 3 4
      WN:=419 512
      XX:=HI,WN
      XX
1 2 3 4 419 512
      A:='CATS'
      B:='NATION'
      $A,B
CATENATION
      10,2,10*4
18 2 40
      18 2 10*4
72 8 40
```

"[" selects elements of the left argument at the locations given by the right argument.

```
      A:=2 17 -3 4
      A[1 1 2
2 2 17
      $'ABCRST'[3 1 6
CAT
```

"f%" performs the Dyadic Scalar Operator f from right to left on all the elements of the right argument. Thus, it treats +%\5 as if it were 1+2+3+4+5.
To find the largest element in a vector, type "%VECTOR.

To write a program (or function), the user must enter EMPL's Definition Mode. Type "&" followed by the new name of the function.

```
      &PAS
>
```

The computer now prompts with a ">" indicating that it is in Definition Mode.
Editing is now like a BASIC editor: Type a line to insert it in the function; type the line number alone to delete the line; type an old line over to replace the old one.
Type "@" to list the function; type "$" to renumber by a specified amount.
For example, the user types in a program to print the first N rows of Pascal's triangle.

```
>10 P:=1
>20 =:PRINT*N)=⍴P:=(0,P)+P,0
>15 PRINT: P
>$ 10
>@
      &PAS
[10]  P:=1
[20]  PRINT: P
[30]  =: PRINT*N)=⍴P:=(0,P)+P,0
>
```

Note that ":=" can be used inside any expression.
"PRINT:" is used as a label in line 20. Upon exiting Definition Mode, PRINT becomes a variable whose value is 20.
"=:" (Control H) is a branch to the line given by the expression which follows it. Branch to a null vector is no branch (computer proceeds to next line), and branch to an undefined line is the end of program.
The user now types a "&" to leave Definition Mode.
To execute a program, type its name.

```
>&
      N:=5
      PAS
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Conditional branches are accomplished by the following:
=:LINE⍴\TEST
If the condition TEST is true, the computer will branch to LINE. Otherwise, the computer will proceed to the next line.

Strings inside double quotes may be used to print comments.

```
      "N=";N
N=5
```

Note that semicolons are used to seperate items, and a semicolon at the end will suppress CRLF.
"⍎" is used to execute a string as if it had been typed in execution mode.

```
      STR:='A:=5*\5'
      ⍎STR
      A
5  10  15  20  25
```

Specific elements of a vector may be assigned values.

```
      A:=\6
      A(3 4):= 17 18
      A
1  2  17  18  5  6
```

When "@" is assigned a value, the value is printed on the terminal.

```
      A:=5*@:=5
5
```

When "$" is assigned a value, the value is printed as a string on the terminal.

```
      A:=$:='TESTING'
TESTING
```

If "@" is used as an operand in an expression, the computer first requests input from the terminal. "$" enables string input to be used. (The input is taken as a literal.)

```
      &TEST
[10]  "WHAT IS YOUR NAME?";
[20]  NAME:=$
[30]  "HI, ";$NAME
[40]  "INPUT AN EXPRESSION"
[50]  EXPR:=@
[60]  "THE ANSWER IS";EXPR
      &
      TEST
WHAT IS YOUR NAME?RACHEL
HI, RACHEL
INPUT AN EXPRESSION
@:5*5
THE ANSWER IS 25
```

User defined monadic or dyadic functions are possible.
Here is an example of a user defined monadic function to take the factorial of the right argument.

```
      &F:=FACT VL
[10]  F:=*%\VL
      &
```

"FACT" may now be used as any EMPL monadic operator can be used.

```
      FACT 3
6
```

When FACT 3 is typed, a function called FACT is searched for.
When it is found, VL is set equal to 3 and the function is
executed. Upon completion of the execution, the computer
uses the variable specified to the left of the assignment
operator "←" as the result.
Here is a user defined dyadic function.

```
      &RS←SZ RHO VTR
[10]  RS←⍳0
[20]  →20*⍳(⍴RS←RS,VTR)<SZ
[30]  RS←RS[⍳SZ
      &
```

This function is used to restructure the vector on the
right such that it has the length indicated on the left.

```
      4 RHO 5 2 3
5 2 3 5
      5 RHO 'ABCDEFG'
ABCD
```

To reenter Definition mode of an old function, just type
its name.

```
      &RHO
```

```
2
```

Parentheses are used to override the normal order of
evaluation.

```
      (3-5)×7
```

```
¯5
```

The following commands may be used inside a program,
or in Execution Mode.

)CLEAR - Clear the workspace.
)FNS - List the names of all functions in the workspace.
)VARS - List the names of all variables in the workspace.
)STOP - Return to execution mode.
)ERASE object - Erase the specified variable or function.
)SI - Display the State Indicator.
)PUR - Clear the State Indicator.
)QUIT - Return to user's monitor
The State Indicator is the stack of the return addresses of all
programs whose execution is pending.
Suppose a program is written, PROG, which calls for the execution
of another one, ASK:

```
      &PROG
[10]  ASK
[20]  "OK"
      .
      .
      .
      &ASK
[10]  HI THERE
      &
      PROG

ASK[10]HI THERE
ERROR 2810
```

The computer prints the program,(or function) and line in which
the error occured.


```
      )SI
ASK[¯1] *
PROG[20] ^
2210 PRES
```

An error has occured in line 10 of ASK. Since the execution
of ASK hasn't been completed, the computer still has the
return address saved. (Line 20 of PROG) Now whenever a →10
instruction be typed, execution will continue at line 20
of PROG.
The program with the "*" to the right of it is the current suspended
program. All "→" (branch) statements refer to this program.
Line ¯1 indicates that it is currently in execution mode.
All programs with "^" to the right of them are pending execution.
The program to return to next is always on the top of the list.
It is a good idea to clear the State Indicator often to
prevent excess garbage from taking up memory. The State Indicator
is automatically cleared when entering Definition Mode, and
after ERROR 5097 (workspace full.)

Here are some sample functions to illustrate EMPL's use.

(1) Generate n prime numbers:

```
      &P←⍳PR END
[10]  P←1↓T←⍳1
[20]  TEST: →0*⍳END<⍴P
[30]  ADD: →ADD*⍳~0=P.T←T+2
[40]  P←P,T
[50]  →TEST
      &
      PR 10
2 3 5 7 11 13 17 19 23 29
      1 + PR 3
3 4 6
```

(2) Compress a vector:

```
      &ANS←SLCT COMP VCTR
[10]  IDX←⍳1,ANS←⍳0
[20]  BK: ANS←ANS,(VCTR[IDX)*⍳⍴SLCT[IDX
[30]  →BK*(IDX←IDX+1)<⍴VCTR
      &
      1 0 1 1 0 COMP ⍳5
1 3 4
```

(3) Sort a group of numbers:
(This function uses COMP)

```
      &ORD←SORT UNS
[10]  ORD←⍳0
[20]  LB: →0*⍳&⍴UNS
[30]  WHICH←UNS=⌊/UNS
[40]  ORD←ORD,WHICH COMP UNS
[50]  UNS←(&WHICH) COMP UNS
[60]  →LB
      &
      SORT ¯2 1 ¯1 0 2
¯2 ¯1 0 1 2
```

(4) Reverse a vector:

```
      &RV1:=REV VC
[10]  VC[(1+ VC)-\^VC
      &
      $REV'HI THERE'
```

FREST IN

(5) Raise the left argument to the right argument.
(They must both be single numbers.)

```
      &E1:=BSE RAIS NUM
[10]  EX1:=*%BSE*&&\NUM
      &
      2 RAIS 5
```

32

EMPL Range = ±32,767

Commands

)CLEAR	Clear Workspace
)FNS	Display function names
)VARS	Display varaible names
)PUR	Clear State Indicator
)ERASE ob	Erase function or variable
)SCN	Clear screen
)SI	Display State indicator
)STOP	Return to execution Mode
)QUIT	Return to monitor

Special Control characters

Control B	Backspace
Control C	Stop
Control F	Forward space
Control H	=:
Control I	::
Control N	>=
Control Y	<=
Control X	Cancel line

Error messages

412	String is too large to execute
885	Object already erased
1082	Unknown operand type in expression
1469	Length of vectors doesn't match
2268	Illegal Operation in reduction
2495	Variable not present in indexing
2529	Length of index doesn't match length of value to be assigned
2559	Index greater than length of variable
2628	Unknown assignment operand type
2690	Function not found
2810	Variable or function not found
3503	Backspace less than beginning of line buffer
3521	Forward spacing greater than end of line buffer
3542	Input over 72 characters
3737	Bad syntax in function definition
3827	Illegal command in definition mode
4058	Illegal redefinition of monadic function header
4083	Illegal redefinition of dyadic function header
5097	Workspace full
5302	Division by zero

* NEW INTERNAL CODE

4) YOUR OUTPUT ROUTINE SHOULD USE A CONVERSION TABLE
TO CONVERT THE LISTED CHARACTER CODE IN 3)
TO THE CORRECT CODE FOR YOUR TERMINAL.

E M P L ERROR MESSAGES

412	STRING IS TOO LARGE TO EXECUTE
885	OBJECT ALREADY ERASED
1082	UNKNOWN OPERAND TYPE IN EXPRESSION
1469	LENGTH OF VECTORS DOESN'T MATCH
2268	ILLEGAL OPERATION IN REDUCTION
2495	VARIABLE NOT PRESENT IN INDEXING
2529	LENGTH OF INDEX DOESN'T MATCH LENGTH OF VALUE TO BE ASSIGNED
2559	INDEX GREATER THAN LENGTH OF VARIABLE
2628	UNKNOWN ASSIGNMENT OPERAND TYPE
2690	FUNCTION NOT FOUND
2810	VARIABLE OR FUNCTION NOT FOUND
3503	BACKSPACE LESS THAN BEGINNING OF LINE BUFFER
3521	FORWARD SPACING GREATER THAN END OF LINE BUFFER
3542	INPUT OVER 72 CHARACTERS
3737	BAD SYNTAX IN FUNCTION DEFINITION
3827	ILLEGAL COMMAND IN DEFINITION MODE
4058	ILLEGAL REDEFIBITION OF MONADIC FUNCTION HEADER
4083	ILLEGAL REDEFINITION OF DYADIC FUNCTION HEADER
5097	WORKSPACE FULL
5302	DIVISION BY ZERO

HOW TO CONVERT EMPL TO RUN ON AN APL TERMINAL

1) STORE 000 020 303
021 LLL
022 HHH

HHHLLL IS THE ADDRESS OF YOUR OUTPUT SUBROUTINE
AS DESCRIBED IN 6) ON PAGE 2.2.
THIS DISABLES THE CONVERSION ON OUTPUT OF CONTROL H,
I, N, AND Y TO =:, :=, >=, AND <=.

2) STORE 000 175 220
001 352 224
004 025 222
004 245 227
006 143 224
010 314 226
010 330 223
012 057 222
014 211 227
014 253 227
015 123 227
016 334 220
020 300 220
021 343 226
021 344 223
021 345 225
021 372 224

THIS ADAPTS EMPL TO NEW INTERNAL CODES DETAILED
IN 3).

3)

EMPL A P L	EMPL ASCII	CODE	YOU TYPE	EMPL A P L	EMPL ASCII	CODE	YOU TYPE
¯	¯	337	¯	≤	=	275	=
!	!	241	!	≠	@	243	@
~	&	246	&	,	,	254	,
?	?	277	?	[	[	333	[
\	\	334	\	/	\	224*	CTRL T
⍴	^	336	^	÷	:=	211	CTRL I
+	+	253	+	⍋	$	244	$
-	-	255	-	¨	"	242	"
×	*	252	*	⍒	%	245	%
⌹	/	257	/	∘	=:	210	CTRL H
⌊	'	226*	CTRL V	(	(	250	(
⌈	"	223*	CTRL S	)	)	251	)
|	.	225*	CTRL W	⍬	@	300	@
<	<	274	<	⍎	$	222*	CTRL R
≤	<=	231	CTRL Y	⍕	'	247	'
>	>	276	>	⍱	-	227*	CTRL V
≥	>=	216	CTRL H	∨	&	220*	CTRL P

E M P L \ 8 0 8 0 LANGUAGE SUMMARY

(Note: EMPL can be run either in ASCII or the APL character set.)

A P L	ASCII	EMPL Category
		Monadic Scalar Operators
_Y	_Y	Minus Y
|Y	|Y	The absolute value of Y
~Y	&Y	Not Y
?Y	?Y	Random from 1 to Y
		Monadic Mixed Operators
⍳Y	\Y	Vector of consecutive integers from 1 to Y
⍴Y	^Y	The length of Y
		Dyadic Scalar Operators
X+Y	X+Y	X plus Y; X or Y
X-Y	X-Y	X minus Y
X×Y	X*Y	X times Y; X and Y
X÷Y	X/Y	X divided by Y
X⌊Y	X'Y	Minimum of X and Y
X⌈Y	X"Y	Maximum of X and Y
X|Y	X.Y	Remainder of X÷Y
X<Y	X<Y	X less than Y
X≤Y	X<=Y	X less than or equal to Y
X>Y	X>Y	X greater than Y
X≥Y	X>=Y	X greater than or equal to Y
X=Y	X=Y	X equal to Y
X≠Y	X#Y	X not equal to Y
		Dyadic Mixed Operators
X,Y	X,Y	Y catenated to X
X⌷Y	X[Y	Elements of X at locations Y
		Composite Operators
f/Y	f$Y	The Operation f performed from right to left on all the elements of Y
		Special Operators and characters
X←Y	X:=Y	Assign Y to X
⍞X	$X	Print X as a string
'⍞TEXT'	"TEXT"	Print literal text
⍎X	%:X	Execute a string
→X	=:X	Branch to X
Y[X]	Y(X)	Indexed variable assignment
⍞	@	Numeric Input/Output
□	$	String Input/Output
'TEXT'	'TEXT'	String vector
(X)	(X)	Expression in parentheses
¯C	-C	Constant negation

A P L	ASCII	EMPL Category
		Definition Mode Commands
□ *N*	@ n	List function
⍝ N	@ n	Renumber
∇	&	Return to execution mode
		Function Definitions
∇*F*	&f	Niladic function; no result
∇*R←F B*	&r:=f b	Monadic function with result
∇*R←A F B*	&r:=a f b	Dyadic function with result
		Commands
)*CLEAR*	)CLEAR	Clear Workspace
)*FNS*	)FNS	Display function names
)*VARS*	)VARS	Display variable names
)*PUR*	)PUR	Clear State indicator
)*ER OB*	)ER ob	Erase function or variable
)*SCR*	)SCR	Clear Screen
)*SI*	)SI	Display State Indicator
)*STOP*	)STOP	Return to execution mode
)*QUIT*	)QUIT	Return to user's monitor

Special Control characters

Control B Backspace
Control C Stop
Control F Forward space
Control H =:,←
Control I :=,←
Control M >=,≥
Control P ∇
Control S ⌊
Control T /
Control U |
Control V ,⍳
Control W ⍵
Control X Cancel line
Control Y <=,≤
Control R ⍴

EMPL Range = ±32767

CORRECTED SAMPLE FUNCTIONS IN APL CHARACTER SET

```
(1) GENERATE N PRIME NUMBERS:
      ∇P←PR END
    [1] P←1↓T←1
    [2] TEST:→0=⍴END≤⍴P
    [3] ADD:→ADD×⍳×/0=P|T←T+2
    [4] P←P,T
    [5] →TEST
      ∇
      PR 10
 2  3  5  7  11  13  17  19  23  29
(2) COMPRESS A VECTOR:
      ∇ANS←SLCT COMP VCTR
    [1] IDX←1,ANS←⍳0
    [2] BK:ANS←ANS,(VCTR[IDX)×⍳SLCT[IDX
    [3] →BK×(IDX←IDX+1)≤⍴VCTR
      ∇
      1 0 1 1 0 COMP ⍳5
1  3  4
(3) SORT A GROUP OF NUMBERS:
    (THIS FUNCTION USES COMP)
      ∇ORD←SORT UNS
    [1] ORD←⍳0
    [2] LB:→0×⍳~⍴UNS
    [3] WHICH←UNS=⌊/UNS
    [4] ORD←ORD,WHICH COMP UNS
    [5] UNS←(~WHICH) COMP UNS
    [6] →LB
      ∇
      SORT ¯2 1 ¯1 0 2
 ¯2  ¯1  0  1  2
      ⍝ SORT 'SORT'
 ORST
(4) REVERSE A VECTOR:
      ∇RV←REV VC
    [1] RV←VC[(1+⍴VC)-⍳⍴VC
      ∇
      ⍝REV'HI THERE'
 EHERT IH
      REV ⍳5
 5  4  3  2  1
(5) RAISE THE LEFT ARGUMENT TO THE RIGHT ARGUMENT.
    (THEY MUST BOTH BE SINGLE NUMBERS.)
      ∇EX←BSE RAIS NUM
    [1] EX←×/BSE*~⍳NUM
      ∇
      2 RAIS 5
 32
```

Sample dialog in EMPL running APL character set:

```
EMPL 1.0
CLEAR WS
      +/⍳10
 55
      ⍝'A'+⍳25
BCDEFGHIJKLMNOPQRSTUVWXYZ
      ∇RV←REV VC
>10 RV←VC[(1+⍴VC)-⍳⍴VC
>□
      ∇RV←REV VC
[10] RV←VC[(1+⍴VC)-⍳⍴VC

>∇
      ⍝ REV 'TESTING '
 GNITSET
      ⍝VC
TESTING
      )SI
REV[¯1] *
2025 FREE
      ∇PASCAL
>P←1

ERROR 3827
>10 P←1
>20 →PRINT×N≥⍴P←(0,P)+P,0
>15 PRINT:P
>⍝10
>□
      ∇PASCAL
[10] P←1
[20] PRINT:P
[30] →PRINT×N≥⍴P←(0,P)+P,0

>∇
      N←6
      PASCAL
 1
 1  1
 1  2  1
 1  3  3  1
 1  4  6  4  1
 1  5  10  10  5  1
      )FNS
REV;PASCAL;
      TEST←⍳3000

ERROR 5097
      TEST

ERROR 2010
```

MISC NOTES

1) NOTE THAT THERE MUST BE A SPACE BETWEEN A '-' AND
A NUMBER WHEN THE MINUS SIGN IS BEING USED AS A DYADIC
OPERATOR AND NOT A CONSTANT NEGATOR. THE SPACE IS NOT
NECESSARY IF A NUMBER IS TO THE LEFT OF THE MINUS SIGN.

```
      A:-5
      A-5
ERROR 2600
      A- 5
0
      A - 5
0
      5-5
0
```

2) WHEN CONTROL X (CANCEL LINE) IS TYPED, THE COMPUTER
PRESENTLY CLEARS THE SCREEN. THE USER MIGHT WISH TO
CHANGE THIS TO PRINT '\' THEN 'CR' FOR A PRINTING
TERMINAL.
A CHANGE MAY BE MADE IN LOCATIONS 015 232 TO 015 237.
EXAMPLE:

```
015 232  076  MVI A,'\'
    233  134
    234  327  RST OUT
    235  367  RST CRLF
    236  000  NOP
    237  000  NOP
```

3) IF THE USER NEEDS TO USE 000 070 FOR INTERRUPT PURPOSES,
THIS MAY BE DONE.
A) INSERT A JMP TO USER'S INTERRUPT SERVICE ROUTINE.

```
000 070  303
    071  XXX
    072  XXX
```

B) USE LOCATION 023 136 AS A NON-INITIALIZATION ENTRY
POINT TO EMPL.

4) THE FIRST 4K BLOCK OF EMPL MAY BE IN PROTECTED MEMORY.

5) IF YOU WISH TO CHANGE THE CHARACTERS ECHOED ON CONTROL
H AND I, STORE THE NEW CHARACTER TO ECHO AS SHOWN BELOW:

```
015 031  XXX  CHARACTER TO ECHO ON CTRL H (=:)
    032  000
    033  000
    034  000
015 045  XXX  CHARACTER TO ECHO ON CTRL I (:=)
    046  000
    047  000
    050  000
```

6) TO DISABLE THE ECHOED CHARACTER ON BACKSPACE (CTRL B),
STORE:

```
015 257  000
    260  000
    261  000
```

7) EMPL CHECKS DURING ALL OPERATIONS FOR AVAILABLE
SPACE IN MEMORY. THE FUNCTIONS AND VARIABLES ARE
WORKING THEIR WAY UP IN MEMORY WHILE THE STATE INDICATOR
(STACK) WORKS ITS WAY DOWN. THE COMPUTER PRESENTLY PRINTS
AN ERROR MESSAGE IF THE TWO ARE CLOSER THAN 256 BYTES.
THE USER MAY EXPERIMENT WITH SMALLER VALUES (THOUGH IT MAY
BE RISKY.)
DECIDE HOW CLOSE YOU WANT TO CHANGE IT TO, THEN NEGATE THE
FIGURE (COMPLEMENT AND ADD 1) AND PUT IT IN MEMORY.

```
023 322  LLL
    323  HHH
```

8) CONTROL CHARACTERS:

```
CTRL A:  SOFTWARE TRANSLATED ON INPUT TO '_' (337)*
CTRL B:  BACKSPACE
CTRL C:  STOP EXECUTION
CTRL D:  SOFTWARE TRANSLATED ON INPUT TO '/' (257)*
CTRL F:  USED TO GO FORWARD IN THE LINE BUFFER WITHOUT
         WRITING OVER OLD DATA.
CTRL G:  SOFTWARE TRANSLATED ON INPUT TO '\' (334)*
CTRL H:  THE INTERNAL CODE FOR AN EMPL OPERATOR.
         TRANSLATED ONLY ON OUTPUT TO '=:'
CTRL I:  TRANSLATED ON OUTPUT TO ':='
CTRL K:  SOFTWARE TRANSLATED ON INPUT TO '{' (333)*
CTRL L:  SOFTWARE TRANSLATED ON INPUT TO '}' (335)*
CTRL N:  TRANSLATED ON OUTPUT TO '>='
CTRL P:  TRANSLATED BY USER ROUTINE TO '▽'**
CTRL R:  TRANSLATED BY USER ROUTINE TO '⍴'**
CTRL S:  TRANSLATED BY USER ROUTINE TO '⌈'**
CTRL T:  TRANSLATED BY USER ROUTINE TO '/'**
CTRL U:  TRANSLATED BY USER ROUTINE TO '|'**
CTRL V:  TRANSLATED BY USER ROUTINE TO '⌊'**
CTRL W:  TRANSLATED BY USER ROUTINE TO '~'**
CTRL X:  CANCEL ENTIRE LINE
CTRL Y:  TRANSLATED ON OUTPUT TO '<='
```

* PROVIDED FOR KEYBOARDS THAT DON'T HAVE THIS CHARACTER.
** NEW INTERNAL CODE FOR AN EMPL OPERATOR WHEN RUNNING
WITH THE APL CHARACTER SET.

9) E M P L \ 8 8 8 8 VERSION 1.0 RESIDES IN THE FIRST
5,632 BYTES OF MEMORY.

10) NOTE THAT THE CORRECT SYNTAX FOR INDEXED VARIABLE
ASSIGNMENT IS'VECTOR(ELEMENTS):=EXPR',NOT'VECTOR(ELEMENTS):=EXPR'
AS STATED IN [illegible]. TO INCREMENT A SINGLE ELEMENT OF A VECTOR,
USE 'VECTOR(ELEMENT):=1+VECTOR(ELEMENT)'.

EMCOS ADAPTED ASCII CHARACTER SET 5/77 A

printing:

@	300	[	333
A	301	\	334
B	302	]	335
C	303	^	336
D	304		337
E	305	`	260
F	306	1	261
G	307	2	262
H	310	3	263
I	311	4	264
J	312	5	265
K	313	6	266
L	314	7	267
M	315	8	270
N	316	9	271
O	317		
P	320	or	215
Q	321	lf	212
R	322		
S	323		
T	324	special	
U	325	functions:	
V	326	ESC	233 esc to EMCOS
W	327	CAN	230 cancel line
X	330	Wc	202 backspace
Y	331	Cc	203 stop
Z	332	Ec	205 halt print
sp	240	Fc	206 forward
!	241		
"	242		
#	243	cursor	
$	244	functions:	
%	245	Pc	220 right
&	246	Qc	221 down
'	247	Rc	222 left
(	250	Sc	223 home
)	251	Tc	224 erase endofline
*	252	Uc	225 up
+	253	Vc	226 erase endoffile
,	254		
-	255		
.	256		
/	257		
:	272		
;	273		
<	274		
=	275		
>	276		
?	277		

translate on input:

Ac	201	
Dc	204	_
Gc	207	/
Kc	213	\
Lc	214	[

translate on output:

Hc	210	=!
Ic	211	!=
Nc	216	>=
Yc	231	<=

XY HEX/DECIMAL/OCTAL/BINARY CONVERSION TABLE

X \ Y	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F	Binary Hex
0000	000	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	Decimal
0	000	001	002	003	004	005	006	007	010	011	012	013	014	015	016	017	Octal
0001	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030	031	Decimal
1	020	021	022	023	024	025	026	027	030	031	032	033	034	035	036	037	Octal
0010	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047	Decimal
2	040	041	042	043	044	045	046	047	050	051	052	053	054	055	056	057	Octal
0011	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062	063	Decimal
3	060	061	062	063	064	065	066	067	070	071	072	073	074	075	076	077	Octal
0100	064	065	066	067	068	069	070	071	072	073	074	075	076	077	078	079	Decimal
4	100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117	Octal
0101	080	081	082	083	084	085	086	087	088	089	090	091	092	093	094	095	Decimal
5	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137	Octal
0110	096	097	098	099	100	101	102	103	104	105	106	107	108	109	110	111	Decimal
6	140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157	Octal
0111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	Decimal
7	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177	Octal
1000	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	Decimal
8	200	201	202	203	204	205	206	207	210	211	212	213	214	215	216	217	Octal
1001	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	Decimal
9	220	221	222	223	224	225	226	227	230	231	232	233	234	235	236	237	Octal
1010	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	Decimal
A	240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257	Octal
1011	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	Decimal
B	260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277	Octal
1100	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	Decimal
C	300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317	Octal
1101	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	Decimal
D	320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337	Octal
1110	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	Decimal
E	340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357	Octal
1111	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	Decimal
F	360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	377	Octal

EMPL

USER'S MANUAL

July 18,1977

Erik T. Mueller
Britton House
Roosevelt, NJ 08555
(609) 448-2605

[illegible]

1) Enter the following checksum loader into memory.
(All locations are split octal.)

```
037 321   061       LXI SP
    322   320
    323   037
    324   016       MOV C,0
    325   000
    326   076       MVI A,020
    327   020
    330   323       OUT 156
    331   156
    332   041       LXI H,0
    333   000
    334   000
    335   315 LOOP  CAL INTP
    336   366
    337   037
    340   373       EI
    341   167       MOV M,A
    342   201       ADD C
    343   117       MOV C,A
    344   043       INX H
    345   174       MOV A,H
    346   376       CPI 026
    347   026
    350   302       JNZ LOOP
    351   335
    352   037
    353   315       CAL INTP
    354   366
    355   037
    356   271       CMP C
    357   302 CSER  JNZ CSER
    360   357
    361   037
    362   363       DI
    363   303 GOOD  JMP GOOD
    364   363
    365   037
    366   333 INTP  IN  156
    367   156
    370   346       ANI 020
    371   020
    372   302       JNZ INTP
    373   366
    374   037
    375   333       IN  157
    376   157
    377   311       RET
```

2) Put tape into cassette recorder.
3) Examine 037 321; Run
4) Start tape
5) When the INTE LED goes on, EMPL is being read into memory.
6) When the address lights change to 037 367 and the INTE LED goes out, the read is complete. If the address lights instead indicate 037 377 with the INTE LED still on, there has been a checksum error and the user must try again.

Implementing EMPL on your system

1) Store the address of the first byte available for the workspace.

```
025 260  LLL  (Low byte of address) — 000
025 261  HHH  (High byte of address) — 052
025 244  LLL — 000
025 245  HHH — 052
```

(Since EMPL resides from 000000 to 025377, the first available byte would normally be 026000 unless the user wishes to put I/O or other subroutines right after EMPL.)

2) Store the address of the first byte available for the workspace + 2.

```
000 105  LLL — 002
000 106  HHH — 052
```

(This would normally be 026002.)

3) Store the address of the last byte available for the workspace.

```
000 001  LLL — 377
000 002  HHH — 137
002 335  LLL — 377
002 336  HHH — 137
016 276  LLL — 377
016 277  HHH — 137
023 137  LLL — 377
023 140  HHH — 137
023 331  LLL — 377
023 332  HHH — 137
```

(In an 8K system, this would be 037 377 if I/O subroutines are right after EMPL or in a monitor somewhere else in memory.)

4) Store the address of your INPUT subroutine.

```
000 011  LLL — 000
000 012  HHH — 026
```

Your INPUT subroutine should get a character from the terminal and put it in the Accumulator (A). Bit 7 should be 1.
The character should not be echoed, as EMPL will use your OUTPUT subroutine to echo it later.
No registers other than A may be changed.
INPUT resides at 026000.

5) Store your data input from keyboard port number.
This is used to check for Control C. (Program interrupt)

```
012 330  xxx — 001
```

6) Store the address of your OUTPUT subroutine.
   015 072 LLL — 006
   015 073 HHH — 026
Your OUTPUT subroutine should print the character in A on your terminal. Bit 7 is 1. No registers including A may be changed.
When a CR (215) is output, your subroutine should also output a LF (if necessary on your terminal.)
Remember that in this case, a CR must be left in A upon exiting the subroutine.
OUTPUT resides at 026006

7) Store the address of your subroutine to clear the screen (if applicable.)
   014 333 LLL — 045 @CRLF instead
   014 334 HHH — 026
If you have no such subroutine,
store 014 333 060.
      334 000

8) Store the character which will cause your terminal to backspace (or the character that you wish to be echoed when Control B is typed.)
This would be a Curser Left on a CRT terminal.
   015 260 xxx — 377
377 translated to 000 on output.

9) Store the character which will cause your terminal to forward space without changing the character on the screen. (This will be echoed when a Control F is typed.)
This would be a Curser Right on a CRT terminal, or simply a space (240) on a printing terminal.
   015 302 xxx — 040 (space)

10) Store the address of your monitor or operating system. (Used for the)QUIT command)
   025 263 LLL — 050
   025 264 HHH — 040

11) Make a new tape of EMPL for future use.

EMPL is now ready to run!

RUNNING EMPL

1) To enter EMPL and create a clear workspace, start execution at 000 000.

2) To exit EMPL, type)QUIT.
To re-enter EMPL leaving the workspace intact, start execution at 000010.

3) To save a workspace on tape:
   a) Type)QUIT. (Quit must be used to exit EMPL, as it stores essential data in workspace for taping.)
   b) Dump memory onto tape starting at the location at 025 260 (L) 025 261 (H), and stepping at the location at 023 027 (L) 023 030 (H).

4) To load a workspace from tape:
   a) Load workspace back in memory from tape.
   b) Start execution at 000010 .

   Note: If the workspace is being loaded into a different part of memory than it was originally in, follow the following directions.
   a) Load workspace back into memory from tape.
   b) Store the new end of tape address at the location at 025 260 (L) 025 261 (H).
   c) Start execution at 000010.

SMPL TAPE MAKER

Here is the program I used to make to SMPL tape:

```
037 000 061        LXI SP
    001 377
    002 037
    003 041        LXI H,0
    004 000
    005 000
    006 076        MVI A,074
    007 074
    010 315        CAL OTP
    011 046
    012 037
    013 076        MVI A,346
    014 346
    015 315        CAL OTP
    016 046
    017 037
    020 016        MVI C,0
    021 000
    022 176 LP,    MOV A,M
    023 315        CAL OTP
    024 046
    025 037
    026 201        ADD C
    027 117        MOV C,A
    030 043        INX H
    031 174        MOV A,H
    032 376        CPI 026
    033 026
    034 302        JNZ LP
    035 022
    036 037
    037 171        MOV A,C
    040 315        CAL OTP
    041 046
    042 037
    043 303 SELF,JMP SELF
    044 043
    045 037
    046 365 OTP,   PUSH PSW
    047 333 WT,    IN   156
    050 156
    051 346        ANI  040
    052 040
    053 302        JNZ WT
    054 047
    055 037
    056 361        POP PSW
    057 323        OUT 157
    060 157
    061 311        RET
```

WS Storage Organization

```
Workspace:   Functions Variables
Functions:   Header (l) (h)  (Prog  Monadic  Dyadic)

                Add to address of H byte
              for address of next object
Prog:        Text
Monadic:     Result (377) Rightvar (377) Text
Dyadic:      Result (377) Leftvar (377) Rightvar (377) Text
Variable:    Header (#) (l) (h) [(L) (H)]^#
                 String or  # of      element HL
                  numeric   elements   of vector
Header:      Name (377) (K00)
Text:        (l)(h)  (#) [Line  (cr)]_0^∞ (377)(377)
                 line   bytes
                number  in line
```

If they meet, WS FULL error.

```
State indicator:   (377) (377) Return address Stacked data
Return address:    PRM: Program called from another program
                   FNR: Program was used as a function
                   REX: Return to execution mode
Stacked data:      Expression data  State indicator
```

K00 = 010 Program
      011 Dyadic function
      012 Monadic function
      004 Variable

```
MPLIO 2A00  3210


 1600                          0010 ********************************************
 1600                          0020 *          I/O FOR EMPL/N.S. DOS INTERFACE   *
 1600                          0030 *          STEVE WILLIAMS 3/15/77 BYPA       *
 1600                          0040 ********************************************
 1600                          0050 *       IN ADDITION TO THIS APPENDATION TO ERIK
UELLER'S
 1600                          0060 *       EMPL, SEVERAL PATCHES HAVE BEEN MADE TO
HE EMPL
 1600                          0070 *       ITSELF. THESE ARE DOCUMENTED IN THE EMPL
MANUAL
 1600                          0090 *       SUPPLIED WITH YOUR COPY. THIS I/O PACKAG
, AND THE
 1600                          0090 *       PATCHES MENTIONED CONSTITUTE THE STANDAR
 BYTE OF
 1600                          0100 *       PALO ALTO EMPL/NORTH STAR CONFIGURATION,
AND ARE
 1600                          0110 *       INTENDED FOR THE SOLE USE OF THE CUSTOME
S WE SUPPLY.
 1600                          0120         ORG   1600H      ;LIVES AT END OF EMPL.
 1600                          0125 PSW     EQU   6          ;KLUDGE FOR SCS.
 1600                          0130 DCHIN   EQU   2010H      ;DOS INPUT CHARACTER.
 1600                          0140 DCHOUT  EQU   200DH      ;DOS OUTPUT CHARACTER.
 1600                          0150 DOS     EQU   202BH      ;DOS RESTART.
 1600                          0160 CR      EQU   0DH
 1600                          0170 LF      EQU   0AH
 1600                          0175 BS      EQU   8
 1600                          0180 *
 1600                          0190 *       INPUT CHARACTER.
 1600                          0200 *
 1600 CD 10 20                 0210 CHIN    CALL  DCHIN      ;GET A CHARACTER FROM DOS.
 1603 F6 80                    0220         ORI   80H        ;SET HIGH BIT.
 1605 C9                       0230         RET              ;RETURN.
 1606                          0240 *
 1606                          0250 *       OUTPUT A CHAR
 1606                          0260 *
 1606 F5                       0270 CHOUT   PUSH  PSW        ;SAVE A
 1607 C5                       0280         PUSH  B          ;SAVE B
 1608 E6 7F                    0290         ANI   7FH        ;STRIP TOP BIT
 160A FE 0D                    0300         CPI   CR         ;CHAR A CR?
 160C CC 1D 16                 0310         CZ    LFEED      ;   YES:DO A LINEFEED FIRST.
 160F FE 7F                    0312         CPI   7FH        ;USE THIS AS A BACKSPACE.
 1611 C2 16 16                 0314         JNZ   NOTBS      ;TRANSLATE.
 1614 3E 08                    0316         MVI   A,BS
 1616 47                       0320 NOTBS   MOV   B,A        ;CHAR TO B FOR DOS.
 1617 CD 0D 20                 0330         CALL  DCHOUT     ;DO OUTPUT.
 161A C1                       0340         POP   B          ;RESTORE REGS.
 161B F1                       0350         POP   PSW
 161C C9                       0360         RET              ;RETURN
 161D 3E 0A                    0370 LFEED   MVI   A,LF       ;GET A LINEFEED.
 161F CD 06 16                 0380         CALL  CHOUT      ;PRINT IT.
 1622 3E 0D                    0390         MVI   A,CR       ;GET CR BACK.
 1624 C9                       0400         RET
 1625                          0410 *
 1625                          0420 *       AT-SIGN ,CRLF. INSTEAD OF CLRSCR.
 1625                          0430 *
 1625 3E 40                    0440 CLR$CR  MVI   A,40H      ;AT-SIGN.
 1627 CD 06 16                 0450         CALL  CHOUT
 162A 3E 0D                    0460         MVI   A,CR       ;CRLF.
 162C C3 06 16                 0470         JMP   CHOUT      ;PRINT AND RETURN FROM THER
E.
 162F                          9998 *       END
```

E M P L , an 8080 APL

Britton House
Roosevelt, N.J. 08555

8K EMPL is a micro APL for the Intel 8080. The current version fits in the first 5,632 bytes of memory. All the special symbols and operators of APL have been adapted to the ASCII character set.

The following description explains EMPL. Examples are used to clarify all operations.

EMPL has two modes: The normal Execution Mode in which all instructions are executed immediately, and the Definition Mode which permits the user to enter functions.
When in Execution Mode, the computer indents five spaces and waits for input.
Type an arithmetic expression and EMPL will respond with the result.

```
      2-6-3
-1
```

(Computer responses are underlined.)
All expressions are evaluated by the right-to-left rule. That is, each operator takes as its right argument everything to its right. For example, in the above problem, EMPL first evaluates 6-3 which equals 3, then it evaluates 2-3 which equals -1.
The range is ±32767-- double-byte integer arithmetic is used.
Variable names may be any length of alphabetic characters. They are assigned values with the ":=" (typed as Control I.)

```
      SPEED:=2059
      SPEED - 59
2000
```

One dimensional arrays are called vectors in EMPL. A variable may be specified to be a vector of any length provided sufficient memory is available.

```
      A:= 5 -4 2 33
      A
5 -4 2 33
      A+1
6 -3 3 34
      A** 3 2 0
20 -12 4 0
```

A Dyadic Operator (such as "+","*","/",etc.) takes two arguments, one on either side of the operator; as compared with a Monadic Operator such as "!" (absolute value), or "?" (random), which takes only one argument to its right.
Constants may be numeric or string vectors. Since the internal representation of both is the same, operations may be performed between the two types. To print a vector as a string, type "$" before it.

```
      X:='STRING'
      X
83 84 82 73 78 71
      $X
STRING
      $X+1
TUSJOH
```


A description of some of EMPL's unique operators follows:
"\" is used to create a vector of consecutive integers between 1 and the right argument.

```
      \5
1 2 3 4 5
```

"^" gives the length of the right argument.

```
      ^'HELLO'
5
      ^\23
23
```

"!" returns the smaller of two numbers; "" returns the greater.

```
      -1'2
-1
      4 1 9"3 2 56
4 2 56
```

"." gives the remainder when the right argument is divided by the left.

```
      2.5
1
```

"<","<="(Control Y),">",">="(Control N),"=", and "#" return a 1 if the condition is true, otherwise they return a 0.

```
      5<4
0
      7 2 4#\3
1 0 1
```

"," chains together two vectors.

```
      HI:= 1 2 3 4
      WN:=419 512
      XX:=HI,WN
      XX
1 2 3 4 419 512
      A:='CAT3'
      B:='NATION'
      $A,B
CAT3NATION
      18,2,10*4
18 2 40
      18 2 10*4
72 8 40
```

"[" selects elements of the left argument at the locations given by the right argument.

```
      A:=2 17 -3 4
      A[1 1 2
2 2 17
      $'ABCRST'[3 1 6
CAT
```

"f%" performs the Dyadic Scalar Operator f from right to left on all the elements of the right argument. Thus, it treats +%\5 as if it were 1+2+3+4+5.
To find the largest element in a vector, type "%VECTOR.

To write a program (or function), the user must enter EMPL's Definition Mode. Type "&" followed by the new name of the function.

```
      &PAS
>
```

The computer now prompts with a ">" indicating that it is in Definition Mode.
Editing is now like a BASIC editor: Type a line to insert it in the function; type the line number alone to delete the line; type an old line over to replace the old one.
Type "@" to list the function; type "$" to renumber by a specified amount.
For example, the user types in a program to print the first N rows of Pascal's triangle.

```
>10 P:=1
>20 =:PRINT*N)=\P:=(0,P)+P,0
>15 PRINT: P
>$ 10
>@
      &PAS
[10] P:=1
[20] PRINT: P
[30] =: PRINT*N)=\P:=(0,P)+P,0
>
```

Note that ":=" can be used inside any expression.
"PRINT:" is used as a label in line 20. Upon exiting Definition Mode, PRINT becomes a variable whose value is 20.
"=:" (Control H) is a branch to the line given by the expression which follows it. Branch to a null vector is no branch (computer proceeds to next line), and branch to an undefined line is the end of program.
The user now types a "&" to leave Definition Mode.
To execute a program, type its name.

```
>&
      N:=5
      PAS
 1
 1  1
 1  2  1
 1  3  3  1
 1  4  6  4  1
```

Conditional branches are accomplished by the following:

```
      =:LINE*\TEST
```

If the condition TEST is true, the computer will branch to LINE. Otherwise, the computer will proceed to the next line.

Strings inside double quotes may be used to print comments.

```
      "N=";N
N=4
```

Note that semicolons are used to separate items, and a semicolon at the end will suppress CRLF.
"%:" is used to execute a string as if it had been typed in execution mode.

```
      STR:='A:=5*\5'
      %:STR
      A
5  10  15  20  25
```

Specific elements of a vector may be assigned values.

```
      A:=\6
      A(3 4):= 17 18
      A
1  2  17  18  5  6
```

When "@" is assigned a value, the value is printed on the terminal.

```
      A:=5*@:=5
5
```

When "$" is assigned a value, the value is printed as a string on the terminal.

```
      A:=$:='TESTING'
TESTING
```

If "@" is used as an operand in an expression, the computer first requests input from the terminal. "$" enables string input to be used. (The input is taken as a literal.)

```
      &TEST
[10] "WHAT IS YOUR NAME?";
[20] NAME:=$
[30] "HI, ";$NAME
[40] "INPUT AN EXPRESSION"
[50] EXPR:=@
[60] "THE ANSWER IS";EXPR
      &
      TEST
WHAT IS YOUR NAME?RACHEL
HI, RACHEL
INPUT AN EXPRESSION
@:5*5
THE ANSWER IS 25
```

User defined monadic or dyadic functions are possible. Here is an example of a user defined monadic function to take the factorial of the right argument.

```
      &F:=FACT VL
[10] F:=*/\VL
      &
```

"FACT" may now be used as any EMPL monadic operator can be used.

```
      FACT 3
6
```

When FACT 3 is typed, a function called FACT is searched for. When it is found, VL is set equal to 3 and the function is executed. Upon completion of the execution, the computer uses the variable specified to the left of the assignment operator ":=" as the result.
Here is a user defined dyadic function.

```
     &RS:=SZ RHO VTR
[10] RS:=\0
[20] =:20*\(^RS:=RS,VTR)<SZ
[30] RS:=RS[\SZ
     &
```

This function is used to restructure the vector on the right such that it has the length indicated on the left.

```
     ^ RHO 5 2 3
5 2 3 5
     $3 RHO 'ABCDEFG'
ABC
```

To reenter Definition mode of an old function, just type its name.

```
     &RHO
2
```

Parentheses are used to override the normal order of evaluation.

```
     (3-5)+7
5
```

The following commands may be used inside a program, or in Execution Mode.

```
)CLEAR  - Clear the workspace.
)FNS    - List the names of all functions in the workspace.
)VARS   - List the names of all variables in the workspace.
)STOP   - Return to execution mode.
)ERASE object - Erase the specified variable or function.
)SI     - Display the State Indicator.
)PUR    - Clear the State Indicator.
)QUIT   - Return to user's monitor
```

The State Indicator is the stack of the return addresses of all programs whose execution is pending.
Suppose a program is written, PROG, which calls for the execution of another one, ASK:

```
     &PROG
[10] ASK
[20] "OK"
      .
      .
      .
     &ASK
[10] HI THERE
     &
     PROG

ASK[10]HI THERE
ERROR 2810
```

The computer prints the program,(or function) and line in which the error occurcd.


```
     )SI
ASK[-1] *
PROG[20] ^
2210 PRES
```

An error has occured in line 10 of ASK. Since the execution of ASK hasn't been completed, the computer still has the return address saved. (Line 20 of PROG) Now whenever a =:0 instruction be typed, execution will continue at line 20 of PROG.
The program with the "*" to the right of it is the current suspended program. All "=:" (branch) statements refer to this program. Line -1 indicates that it is currently in execution mode. All programs with "^" to the right of them are pending execution. The program to return/to next is Always on the top of the list. It is a good idea to clear the State indicator often to prevent excess garbage from taking up memory. The State Indicator is automatically cleared when entering Definition Mode, and after ERROR 5097 (workspace full.)

Here are some sample functions to illustrate EMPL's use.

(1) Generate n prime numbers:

```
     &P:=PR END
[10] P:=1+T:=1
[20] TEST: =:0*\END<=^P
[30] ADD: =:ADD*\+/0=P.T:=T+2
[40] P:=P,T
[50] =:TEST
     &
     PR 10
2 3 5 7 11 13 17 19 23 29
       1 + PR 3
3 4 6
```

(2) Compress a vector:

```
     &ANS:=SLCT COMP VCTR
[10] IDX:=1,ANS:=\0
[20] EK: ANS:=ANS,(VCTR[IDX)*\SLCT[IDX
[30] =:EK*(IDX:=IDX+1)<=^VCTR
     &
     1 0 1 1 0 COMP \5
1 3 4
```

(3) Sort a group of numbers:
(This function uses COMP)

```
     &ORD:=SORT UNS
[10] ORD:=\0
[20] LB: =:0*\&^UNS
[30] WHICH:=UNS='/UNS
[40] ORD:=ORD,WHICH COMP UNS
[50] UNS:=(&WHICH) COMP UNS
[60] =:LB
     &
     SORT -2 1 -1 0 2
-2 -1 0 1 2
```

(4) Reverse a vector:

```
      &RV1:=REV VC
    [10] VC[(1+ VC)-\\VC
      &
      $REV'HI THERE'
```

ERESET 14

(5) Raise the left argument to the right argument.
(They must both be single numbers.)

```
      &E1:=BSE RAIS NUM
    [10] E1:=*%BSE*&&\NUM
      &
      2 RAIS 5
```

32

EMPL Range = ±32,767

Commands

)CLEAR	Clear Workspace
)FNS	Display function names
)VARS	Display varaible names
)PUR	Clear State Indicator
)ERASE ob	Erase function or variable
)SCN	Clear screen
)SI	Display State indicator
)STOP	Return to execution Mode
)QUIT	Return to monitor

Special Control characters

Control B	Backspace
Control C	Stop
Control F	Forward space
Control H	=:
Control I	:=
Control N	>=
Control Y	<=
Control X	Cancel line

Error messages

412	String is too large to execute
885	Object already erased
1082	Unknown operand type in expression
1469	Length of vectors doesn't match
2268	Illegal Operation in reduction
2495	Variable not present in indexing
2529	Length of index doesn't match length of value to be assigned
2559	Index greater than length of variable
2628	Unknown assignment operand type
2690	Function not found
2810	Variable or function not found
3503	Backspace less than beginning of line buffer
3521	Forward spacing greater than end of line buffer
3542	Input over 72 characters
3737	Bad syntax in function definition
3827	Illegal command in definition mode
4058	Illegal redefinition of monadic function header
4083	Illegal redefinition of dyadic function header
5097	Workspace full
5302	Division by zero

* NEW INTERNAL CODE

4) YOUR OUTPUT ROUTINE SHOULD USE A CONVERSION TABLE
TO CONVERT THE LISTED CHARACTER CODE IN 3)
TO THE CORRECT CODE FOR YOUR TERMINAL.

E M P L ERROR MESSAGES

412	STRING IS TOO LARGE TO EXECUTE
885	OBJECT ALREADY ERASED
1082	UNKNOWN OPERAND TYPE IN EXPRESSION
1469	LENGTH OF VECTORS DOESN'T MATCH
2268	ILLEGAL OPERATION IN REDUCTION
2495	VARIABLE NOT PRESENT IN INDEXING
2529	LENGTH OF INDEX DOESN'T MATCH LENGTH OF VALUE TO BE ASSIGNED
2559	INDEX GREATER THAN LENGTH OF VARIABLE
2628	UNKNOWN ASSIGNMENT OPERAND TYPE
2690	FUNCTION NOT FOUND
2810	VARIABLE OR FUNCTION NOT FOUND
3503	BACKSPACE LESS THAN BEGINNING OF LINE BUFFER
3521	FORWARD SPACING GREATER THAN END OF LINE BUFFER
3542	INPUT OVER 72 CHARACTERS
3737	BAD SYNTAX IN FUNCTION DEFINITION
3827	ILLEGAL COMMAND IN DEFINITION MODE
4058	ILLEGAL REDEFINITION OF MONADIC FUNCTION HEADER
4083	ILLEGAL REDEFINITION OF DYADIC FUNCTION HEADER
5097	WORKSPACE FULL
5302	DIVISION BY ZERO

HOW TO CONVERT EMPL TO RUN ON AN APL TERMINAL

1) STORE 000 020 303
021 LLL
022 HHH

HHLLL IS THE ADDRESS OF YOUR OUTPUT SUBROUTINE
AS DESCRIBED IN 6) ON PAGE 2.2.
THIS DISABLES THE CONVERSION ON OUTPUT OF CONTROL H,
I, N, AND Y TO =:, :=, >=, AND <=.

2) STORE 000 175 220
001 352 224
004 025 222
004 245 227
006 143 224
010 314 226
010 330 223
012 057 222
014 211 227
014 253 227
015 123 227
016 334 220
020 300 220
021 343 226
021 344 223
021 345 225
021 372 224

THIS ADAPTS EMPL TO NEW INTERNAL CODES DETAILED
IN 3).

3)

EMPL APL	EMPL ASCII	CODE	YOU TYPE	EMPL APL	EMPL ASCII	CODE	YOU TYPE
¯	-	337	-	≠	=	275	=
¡	!	241	!	≠	@	243	@
~	&	246	&	.	.	254	.
?	?	277	?	[	[	333	[
\	\	334	\	/	%	224*	CTRL T
⍵	^	336	^	÷	:=	211	CTRL I
+	+	253	+	⍵	$	244	$
-	-	255	-	¨	"	242	"
×	*	252	*	∈	%	245	%
⍵	/	257	/	÷	=:	210	CTRL H
⌊	'	226*	CTRL V	(	(	250	(
⌈	"	223*	CTRL S	)	)	251	)
∣	.	225*	CTRL U	⍬	@	300	@
<	<	274	<	⍷	$	222*	CTRL R
≤	<=	231	CTRL Y	'	'	247	'
>	>	276	>	¯	_	227*	CTRL W
≥	>=	216	CTRL N	∨	&	220*	CTRL P

EMPL\8080 LANGUAGE SUMMARY

(Note: EMPL can be run either in ASCII or the APL character set.)

APL	ASCII	EMPL Category
		Monadic Scalar Operators
-Y	-Y	Minus Y
|Y	|Y	The absolute value of Y
~Y	&Y	Not Y
?Y	?Y	Random from 1 to Y
		Monadic Mixed Operators
⍳Y	\Y	Vector of consecutive integers from 1 to Y
⍴Y	^Y	The length of Y
		Dyadic Scalar Operators
X+Y	X+Y	X plus Y; X or Y
X-Y	X-Y	X minus Y
X×Y	X*Y	X times Y; X and Y
X÷Y	X/Y	X divided by Y
X⌊Y	X'Y	Minimum of X and Y
X⌈Y	X"Y	Maximum of X and Y
X|Y	X.Y	Remainder of X÷Y
X<Y	X<Y	X less than Y
X≤Y	X<=Y	X less than or equal to Y
X>Y	X>Y	X greater than Y
X≥Y	X>=Y	X greater than or equal to Y
X=Y	X=Y	X equal to Y
X≠Y	X#Y	X not equal to Y
		Dyadic Mixed Operators
X,Y	X,Y	Y catenated to X
X⌷Y	X[Y	Elements of X at locations Y
		Composite Operators
f/Y	f%Y	The Operation f performed from right to left on all the elements of Y
		Special Operators and characters
X←Y	X:=Y	Assign Y to X
⍞X	$X	Print X as a string
"TEXT"	"TEXT"	Print literal text
⍎X	%:X	Execute a string
→X	=:X	Branch to X
Y(X)	Y(X)	Indexed variable assignment
⍞	@	Numeric Input/Output
⍞	$	String Input/Output
'TEXT'	'TEXT'	String vector
(X)	(X)	Expression in parentheses
¯C	-C	Constant negation

APL	ASCII	EMPL Category
		Definition Mode Commands
∇ #	@ n	List function
∇ #	$ n	Renumber
∇	&	Return to execution mode
		Function Definitions
∇F	&f	Niladic function; no result
∇R←F B	&r:=f b	Monadic function with result
∇R←A F B	&r:=a f b	Dyadic function with result
		Commands
)CLEAR	]CLEAR	Clear Workspace
)FNS	]FNS	Display function names
)VARS	]VARS	Display variable names
)PUR	]PUR	Clear State indicator
)ER OB	]ER ob	Erase function or variable
)SCR	]SCR	Clear Screen
)SI	]SI	Display State Indicator
)STOP	]STOP	Return to execution mode
)QUIT	]QUIT	Return to user's monitor

Special Control characters

Control B	Backspace
Control C	Stop
Control F	Forward space
Control H	=:,-
Control I	:=,+
Control M	>=,z
Control P	∇
Control S	[
Control T	/
Control U	|
Control V	¯
Control W	⍀
Control X	Cancel line
Control Y	<=,s
Control R	⍓

EMPL Range = ±32767

CORRECTED SAMPLE FUNCTIONS IN APL CHARACTER SET

```
(1) GENERATE N PRIME NUMBERS:
        ∇P←PR END
    [1] P←1↓T←⍳1
    [2] TEST:→0=⍴⍳END≤⍴P
    [3] ADD:→ADD=⍴⍳+/0=P|T←T+2
    [4] P←P,T
    [5] →TEST
        ∇
        PR 10
     2  3  5  7  11  13  17  19  23  29
(2) COMPRESS A VECTOR:
        ∇ANS←SLCT COMP VCTR
    [1] IDX←1,ANS←⍳0
    [2] BK:ANS←ANS,(VCTR[IDX]=⍳SLCT[IDX
    [3] →BK×(IDX←IDX+1)≤⍴VCTR
        ∇
        1 0 1 1 0 COMP ⍳5
    1  3  4
(3) SORT A GROUP OF NUMBERS:
    (THIS FUNCTION USES COMP)
        ∇ORD←SORT UNS
    [1] ORD←⍳0
    [2] LB:→0=⍴⍳~⍴UNS
    [3] WHICH←UNS=⌊/UNS
    [4] ORD←ORD,WHICH COMP UNS
    [5] UNS←(~WHICH) COMP UNS
    [6] →LB
        ∇
        SORT ¯2 1 ¯1 0 2
    ¯2  ¯1  0  1  2
        $ SORT 'SORT'
    ORST
(4) REVERSE A VECTOR:
        ∇RV←REV VC
    [1] RV←VC[(1+⍴VC)-⍳⍴VC
        ∇
        $REV'HI THERE'
    EHERT IH
        REV ⍳5
     5  4  3  2  1
(5) RAISE THE LEFT ARGUMENT TO THE RIGHT ARGUMENT.
    (THEY MUST BOTH BE SINGLE NUMBERS.)
        ∇EX←BSE RAIS NUM
    [1] EX←×/BSE×⍤¯1⍳NUM
        ∇
        2 RAIS 5
    32
```

Sample dialog in EMPL running APL character set:

```
EMPL 1.0
CLEAR WS
      +/⍳10
 55
      $'A'+⍳25
BCDEFGHIJKLMNOPQRSTUVWXYZ
      ∇RV←REV VC
>10 RV←VC[1+⍴VC)-⍳⍴VC
>□
      ∇RV←REV VC
[10] RV←VC[(1+⍴VC)-⍳⍴VC

>∇
      $ REV 'TESTING '
 GNITSET
      $VC
TESTING
      )SI
REV[¯1] *
2025 FREE
      ∇PASCAL
>P←1

ERROR 3027
>10 P←1
>20 →PRINT×N≥⍴P←(0,P)+P,0
>15 PRINT:P
>$1C
>□
      ∇PASCAL
[10] P←1
[20] PRINT:P
[30] →PRINT×N≥⍴P←(0,P)+P,0

>∇
      N←6
      PASCAL
 1
 1  1
 1  2  1
 1  3  3  1
 1  4  6  4  1
 1  5  10  10  5  1
      )FNS
KEY,PASCAL,
      TEST←⍳3000

ERROR 5097
      TEST

ERROR 2810
```

MISC NOTES

1) NOTE THAT THERE MUST BE A SPACE BETWEEN A '-' AND A NUMBER WHEN THE MINUS SIGN IS BEING USED AS A DYADIC OPERATOR AND NOT A CONSTANT NEGATOR. THE SPACE IS NOT NECESSARY IF A NUMBER IS TO THE LEFT OF THE MINUS SIGN.

```
      A←5
      A-5
ERROR 2690
      A- 5
0
      A - 5
0
      5-5
0
```

2) WHEN CONTROL X (CANCEL LINE) IS TYPED, THE COMPUTER PRESENTLY CLEARS THE SCREEN. THE USER MIGHT WISH TO CHANGE THIS TO PRINT '\' THEN 'CR' FOR A PRINTING TERMINAL.
A CHANGE MAY BE MADE IN LOCATIONS 015 232 TO 015 237.
EXAMPLE:

```
015 232  076  MVI A,'\'
    233  134
    234  327  RST OUT
    235  367  RST CRLF
    236  000  NOP
    237  000  NOP
```

3) IF THE USER NEEDS TO USE 000 070 FOR INTERRUPT PURPOSES, THIS MAY BE DONE.
A) INSERT A JMP TO USER'S INTERRUPT SERVICE ROUTINE.

```
000 070  303
    071  XXX
    072  XXX
```

B) USE LOCATION 023 136 AS A NON-INITIALIZATION ENTRY POINT TO EMPL.

4) THE FIRST 9K BLOCK OF EMPL MAY BE IN PROTECTED MEMORY.

5) IF YOU WISH TO CHANGE THE CHARACTERS ECHOED ON CONTROL N AND I, STORE THE NEW CHARACTER TO ECHO AS SHOWN BELOW:

```
015 031  XXX  CHARACTER TO ECHO ON CTRL N (↑:)
    032  000
    033  000
    034  000
015 045  XXX  CHARACTER TO ECHO ON CTRL I (:=)
    046  000
    047  000
    050  000
```

6) TO DISABLE THE ECHOED CHARACTER ON BACKSPACE (CTRL B), STORE:

```
015 257  000
    260  000
    261  000
```

7) EMPL CHECKS DURING ALL OPERATIONS FOR AVAILABLE SPACE IN MEMORY. THE FUNCTIONS AND VARIABLES ARE WORKING THEIR WAY UP IN MEMORY WHILE THE STATE INDICATOR (STACK) WORKS ITS WAY DOWN. THE COMPUTER PRESENTLY PRINTS AN ERROR MESSAGE IF THE TWO ARE CLOSER THAN 256 BYTES. THE USER MAY EXPERIMENT WITH SMALLER VALUES (THOUGH IT MAY BE RISKY.)
DECIDE HOW CLOSE YOU WANT TO CHANGE IT TO, THEN NEGATE THE FIGURE (COMPLEMENT AND ADD 1) AND PUT IT IN MEMORY.

```
023 322  LLL
    323  HHH
```

8) CONTROL CHARACTERS:

CTRL A: SOFTWARE TRANSLATED ON INPUT TO '_' (137)•
CTRL B: BACKSPACE
CTRL C: STOP EXECUTION
CTRL D: SOFTWARE TRANSLATED ON INPUT TO '/' (257)•
CTRL F: USED TO GO FORWARD IN THE LINE BUFFER WITHOUT WRITING OVER OLD DATA.
CTRL G: SOFTWARE TRANSLATED ON INPUT TO '\' (134)•
CTRL N: THE INTERNAL CODE FOR AN EMPL OPERATOR. TRANSLATED ONLY ON OUTPUT TO '←:'
CTRL I: TRANSLATED ON OUTPUT TO ':='
CTRL K: SOFTWARE TRANSLATED ON INPUT TO '{' (333)•
CTRL L: SOFTWARE TRANSLATED ON INPUT TO '}' (335)•
CTRL M: TRANSLATED ON OUTPUT TO '>='
CTRL P: TRANSLATED BY USER ROUTINE TO '▼'••
CTRL R: TRANSLATED BY USER ROUTINE TO '⍎'••
CTRL S: TRANSLATED BY USER ROUTINE TO '⌈'••
CTRL T: TRANSLATED BY USER ROUTINE TO '/'••
CTRL U: TRANSLATED BY USER ROUTINE TO '|'••
CTRL V: TRANSLATED BY USER ROUTINE TO '⌊'••
CTRL W: TRANSLATED BY USER ROUTINE TO '⍣'••
CTRL X: CANCEL ENTIRE LINE
CTRL Y: TRANSLATED ON OUTPUT TO '<='

• PROVIDED FOR KEYBOARDS THAT DON'T HAVE THIS CHARACTER.
•• NEW INTERNAL CODE FOR AN EMPL OPERATOR WHEN RUNNING WITH THE APL CHARACTER SET.

9) E M P L \ 0 0 0 0 VERSION 1.0 RESIDES IN THE FIRST 5,632 BYTES OF MEMORY.

10) NOTE THAT THE CORRECT SYNTAX FOR INDEXED VARIABLE ASSIGNMENT IS'VECTOR(ELEMENTS):←EXPR',NOT'VECTOR[ELEMENTS]:←EXPR' AS STATED IN GOS. TO INCREMENT A SINGLE ELEMENT OF A VECTOR, USE 'VECTOR(ELEMENT):←1+VECTOR[ELEMENT'.

EMCOS ADAPTED ASCII CHARACTER SET 5/77 A

printing:				translate on input:	
@	300	[	333	A^c	201 _
A	301	\	334	D^c	204 /
B	302	]	335	G^c	207 \
C	303	^	336	K^c	213 [
D	304	_	337	L^c	214]
E	305	`	260		
F	306	1	261		
G	307	2	262		
H	310	3	263		
I	311	4	264	translate on output:	
J	312	5	265	M^c 210 =|	
K	313	6	266	I^c 211 !|	
L	314	7	267	N^c 216 >|	
M	315	8	270	Y^c 231 <|	
N	316	9	271		
O	317				
P	320	cr	215		
Q	321	lf	212		
R	322				
S	323				
T	324	special			
U	325	functions:			
V	326	ESC	233 esc to EMCOS		
W	327	CAN	230 cancel line		
X	330	B^c	202 backspace		
Y	331	C^c	203 stop		
Z	332	E^c	205 halt print		
sp	240	F^c	206 forward		
!	241				
"	242				
#	243	oursor			
$	244	functions:			
%	245	P^c	220 right		
&	246	Q^c	221 down		
'	247	R^c	222 left		
(	250	S^c	223 home		
)	251	T^c	224 erase endofline		
*	252	U^c	225 up		
+	253	V^c	226 erase endoffile		
,	254				
-	255				
.	256				
/	257				
:	272				
;	273				
<	274				
=	275				
>	276				
?	277				

XY

HEX/DECIMAL/OCTAL/BINARY CONVERSION TABLE

Y / X	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 A	1011 B	1100 C	1101 D	1110 E	1111 F	Binary / Hex
0000	000	001	002	003	004	005	006	007	008	009	010	011	012	013	014	015	Decimal
0	000	001	002	003	004	005	006	007	010	011	012	013	014	015	016	017	Octal
0001	016	017	018	019	020	021	022	023	024	025	026	027	028	029	030	031	Decimal
1	020	021	022	023	024	025	026	027	030	031	032	033	034	035	036	037	Octal
0010	032	033	034	035	036	037	038	039	040	041	042	043	044	045	046	047	Decimal
2	040	041	042	043	044	045	046	047	050	051	052	053	054	055	056	057	Octal
0011	048	049	050	051	052	053	054	055	056	057	058	059	060	061	062	063	Decimal
3	060	061	062	063	064	065	066	067	070	071	072	073	074	075	076	077	Octal
0100	064	065	066	067	068	069	070	071	072	073	074	075	076	077	078	079	Decimal
4	100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117	Octal
0101	080	081	082	083	084	085	086	087	088	089	090	091	092	093	094	095	Decimal
5	120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137	Octal
0110	096	097	098	099	100	101	102	103	104	105	106	107	108	109	110	111	Decimal
6	140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157	Octal
0111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	Decimal
7	160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177	Octal
1000	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	Decimal
	200	201	202	203	204	205	206	207	210	211	212	213	214	215	216	217	Octal
1001	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	Decimal
9	220	221	222	223	224	225	226	227	230	231	232	233	234	235	236	237	Octal
1010	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	Decimal
A	240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257	Octal
1011	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	Decimal
B	260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277	Octal
1100	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	Decimal
C	300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317	Octal
1101	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	Decimal
D	320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337	Octal
1110	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	Decimal
E	340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357	Octal
1111	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	Decimal
F	360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	377	Octal