EXPRESS BASIC AND COMPILER FOR MZ80K

A cassette based compiler system available for the SHARP MZ80K abd MZ700 series computers, comprising:

EXPRESS BASIC	An interpreter which itself is some three times faster than standard SHARP BASIC
EXPRESS COMPILER	Providing the option of running another twenty times faster, by converting BASIC programs automatically into machine code.

The MZ700 version can also run on the MZ80A...

Operation is simple.

Programs are written in BASIC in the usual way, Instructions are given to compile and in a few seconds are converted to machine code and ready for recording to cassette.
Compiling BASIC code to machine code is quick and easy.
Problems with your machine code if a fault is found is greatly simplified by correcting the original BASIC code and re-compiling.

EXPRESS BASIC

EXPRESS BASIC is an integer BASIC (Whole numbers: +- 32767) and has limited string facilities plus other commands not included.
EXPRESS BASIC has a different structure to SHARP BASIC and so are incompatable.
Variable names may be any length and are fully discriminated.

Interpreter occupies $1200 to $33FF.
Cold Start = $1200
Warm Start = $1203 If in MON enter 'B' followed by <CR>

COMMANDS

IF...THEN	FOR...NEXT	STEP	ON	GOTO (JMP)
GOSUB (SUB)	RETURN (RET)	REM (')	INPUT	STOP
END	BYE	POKE	TO	MUSIC
TEMPO	ABS(	SGN(	PEEK(	NEW
RUN*	LOAD*	LIMIT*	SAVE*	VERIFY*
INP(	OUT	CLR*	CURSOR H,V	
: (Seperator)				

* = DISCARDED BY COMPILER

ADRS(x) A=ADRS(B) subscribes to 'A' the decimal value of the memory location of variable 'B'

APPEND Adds a basic program from tape to another already in memory. The second program must be numbered higher than the first.

Arrays Similer to SHARPS BASIC
eg: X(Y)=Z, Where! X must be defined in memory, being the storage start address.
No limit to array size but numeric only.
Use of square brackets '[ ]' provides for single byte arrays 0-255.
Use X(-Y) or X[-Y] provides for a secondary array running down in memory.

BELL Sounds bell! (CALL 62)

CALL Calls a machine code routine in memory!

CURSOR H,V Positions cursor on TEXT screen
Where: H=horizontal (0 to 39)
V=vertical (0 to 24)

**CP$(** W=CP$($XXXX,$YYYY,"A") compares strings
$XXXX = start address string 1
$YYYY = start address string 2
If identical before chr 'A' in first string W=1 else W=0

CPI(W=CPI($XXXX,"AB..") 'W'=1 if "AB.." is present at location $XXXX, else 'W'=0

DELETE X,Y Deletes BASIC 0ines 'X' through 'Y'

DEC(A=DEC(A) is equivalent to A=A-1 but faster!

INC(A=INC(A) is equivalent to A=A+1 but faster!

GET@ H1,V1,H2,V2,$addr Stores at location $addr onwards the displaycodes of screen rectangle defined by co-ordinates H1V1, H2V2. H and V are 0-39, 0-24

PUT@ H1,V1,H2,V2,$addr Takes the characters stored as display codes from location $addr onwards and displays on screen in the rectangle defined by co-ordinates H1V1 and H2V2. h=039, V=0-24

INPUT Accepts numeric input only - See LINPUT

INKEY A = INKEY, as GET but does not wait.

IF...GOTO Must use: IF...GOTO, IF...GOSUB, IF...THEN (No line No: with THEN)
eg: IF A=10 GOTO 100
IF A=10 GOSUB 100
IF A=10 THEN B=5 or IF A=10 B=5

LINPUT LINPUT A accepts a user input and stores at memory location A as a string of 41 or 81 ASCII char's, shorter inputs are supplemented with '0D'

LINE H1,V1,H2,V2,Z If Z=1 draws a line from H1V1 to H2V2
If Z=0 deletes line from H1V1 to H2V2 H and V are 0-79, 0-49

LIST X,Y Lists from X to Y Basic program.
Also: 'LIST X' 'LIST Y' 'LIST X,'

MON Passes control to EXPRESS MONITOR routine.

BYE Passes control to SHARP MONITOR.

MOD(X,Y) Integer division, A=MOD(X,Y) gives X-Y(X/Y)

OR, XOR,AND Boolean operations. Enclosed expressions in brackets.
As in SHARP BASIC, + - * may be used.
eg: IF (A=5)*(B=7) THEN 'AND'
IF (A=5)+(B=7) THEN 'OR'

POP X Takes variable X from stack.

PUSH X Puts variable X on top of stack.

PRMODE X Sets printout mode. X=0 output to screen.
X=1 output to printer.
X=2 output to screen and printer.

PLOT H,V,Z Equivalent to SET and RESET.
H=0-79 horizontal, V=0-49 vertical.
Z=1 'SET' Z=0 'RESET'

PRINT or ? Prints without <CR>. Use'/' for ,CR> and chr',' to searate items in multiple print statements
eg: ? a,/b,/c ?"abc",/"def"
? a,b,c ? "abc","def",// etc...

CP$(	W=CP$($XXXX,$YYYY,"A") compares strings $XXXX = start address string 1 $YYYY = start address string 2 If identical before chr 'A' in first string W=1 else W=0
PRINT%X	Permits numbers to be tabulated, if A=5 and B=500 then: ?%0,A,/B gives 5 ?%3,A,/B gives 5 500 500
?# X	Converts to hex, where X is number of digits (2 or 4) ?#2 60 gives 3C, ?#4 60 gives 003C
?$	Gives hex to decimal conversion eg: ?$3C gives 60
? TAB(X),"A"	Use instead CURH X:?"A".
?& 65	Equivalent of CHR$. eg: ?&65 gives "A"
PRINT (string)	PUSH $addr:CALL $1256, prints the string at location $addr to the screen at current screen location until $0D encounted. see STA$
RND(X)	Generates random number between 0 and X.
REPEAT...UNTIL	Repeats intervening section until condition met. eg: REPEAT:G=GET:?G:UNTIL G=49 would keep getting and printing G until the user pressed '!' chr (ASCII 49).
STA$($ADDR,"a...")	Stores string "a..." at address $addr onwards. If the string is to be printed using CALL $1256, then $0D is used to end the string.
SOUND X	Sounds continous note where X is between 256 (high) and 32767 (low). SOUND 0 = OFF. Execution continues whilst note sounds. The normal SHARP access to the sound generator POKEs to 4513 and 4514, with CALL 68 / CALL 71 to switch on/off is also available, but runs at double speed on MZ-700.
SHIFT/BREAK	Care needed on MZ80A and MZ-700 if used during SOUND or INPUT control not returned to keyboard - use J1203 in preferance.
SPEEK(X,Y)	Screen PEEK. eg: A=SPEEK(X,Y) Returns display code in A at screen location X,Y where X=0to39 Y=0to24
SYSTEM	Displays BASIC program information. TEXT START, TESXT END, MEMORY END
$	Indicates number following is hex notation.
TEXT$addr	Alters location at which source code (BASIC program) is put from normal. Normal BASIC program starts at $4000 to $XXXX.
TIME	Use TIME=0 and ?TIME in place of TI$="000000" and ? TI$. NB: Clock counts in seconds and starts running when TIME= has been used.
VLIST	Lists all variables and their values.
COLOUR	Two digit number. $XY where X,Y are char and background colours 0-7 Any time after establishing colours to be used by 'PUSH $XY:CALL $1266', 'CALL $1274' will colour whole screen - blue/white will return if the screen is allowed to advance. POKE colour code to location 2048 above screen addr to change colour combination at any individual screen location. MZ-700 ONLY!
FOR	'FOR' loop instruction ending with 'NEXT' eg: FOR T = 0 TO 20 : NEXT T
TO	User must insert spaces before and after.
STEP	Negative STEP not permitted.

** EXPRESS COMPILER **

The compiler converts to machine code in part by identifying and calling run-time routines which are provided at the start of the BASIC interpreter and in part by constructing code directly. The latter code is first placed in a temporary location for test running. This is done so as to preserve intact the BASIC interpreter and program so that changes can be made if any problems arise in compilation. The program is then re-compiled and the code placed immediately above the run-time routines in the space formerly occupied by the interpreter, so that the completed machine code program requires the minimum amount of memory for saving to cassette.

The normal procedure is as below. On completion of compilation there is a display of data which indicates where in memory the code relating to each BASIC line is located, where all variables are stored and the start and end address of the generated code (apart from run-time routines $1200-$1DFF) Hard copy is provided if a printer is connected and PRMODE 2 is called before loading the compiler. In the event of any fault being met, compilation is halted with the defective BASIC line indicated. Answering the prompt 'M:' with 'B <CR>.' returns to BASIC. The addresses suggested below for location of source code (BASIC) and object code (machine code) and work area may of course be varied by experienced programmers to suit their own purposes.

1) Load EXPRESS BASIC, enter BYE <CR>
   Load EXPRESS COMPILER and press <CR> then 'B <CR> to return to BASIC.

2) Enterprogram or load from cassette, test run and save.

3) Call EXPRESS COMPILER by CALL $9800

4) Compile to a temporary location and test run the m/c code by responding to prompts as under:

```
SOURCE PROGRAM ADRS:  $4000   <CR>
OBJECT PROGRAM  ADRS:  $7000   <CR>
WORKING AREA       ADRS:  $3400   <CR>

M: J7000  <CR>
```

5) Compile to final location by responding to prompts as below:

```
M: J9800

SOURCE PROGRAM ADRS:  $4000   <CR>
OBJECT PROGRAM  ADRS:  $1E00   <CR>     Attached to run-time!
WORKING AREA       ADRS:  $3400   <CR>
```

6) Note(the address $ZZZZ for the end o m/c which is displayed on screen as:
OBJECT SIZE: XXX(YYYY - $ZZZZ) and save the compiled program to cassette by responding to prompts under:

```
M: S <CR>
FILENAME: 'NAME OF PROGRAM'  <CR>
START:    1200   <CR>
END:      ZZZZ   <CR>
AUTO:     1200   <CR>
RECORD.PLAY
```

The compiler occupies $9800 - $BAFF. Access from BASIC by CALL $9800 and from monitor by J9800.

** EXPRESS MONITOR **

The monitor permits writing and editing of programs by direct entry of hex m/c. The routines for this are contained in the interpreter and so always available when BASIC is loaded.
Monitor can be called from BASIC by the 'MON' command and direct entry from the compiler on completion of compilation of if any fault is met during compile run.

On entering monitor the prompt 'M:' is displayed and can be answered by one letter commands:

'B' = Return to BASIC interpreter
'D' = DUMP memory
'E' = EDIT memory
'J' = JUMP to memory
'S' = SAVE memory

* 'D' DUMP memory *

Entering 'D XXXX <CR>' where

XXXX = An hex address will dump the memory contents on the screen
8 bytes per line with ASCII char's.
Pressing <spc> bar will pause listing and SHIFT-BREAK will halt listing.

* 'E' EDIT memory *

After calling 'd' to display block code, enter 'E <CR>' and the code can be modified by the user of cursor controls. Press <CR> after each line modified. Enter '; <CR>' to exit.

To write completely new code from hex XXXX addr, in response to prompt 'M:' enter 'E <CR>' then enter your addr '$XXXX <CR>'. The addr will be displayed awaiting your new code entry. There is no need to leave spaces between each pair of digits. A new prompt appears automatically when reaching the end of any display code being modified, unless ';' has been entered.

* 'J' JUMP to memory *

Entering 'JXXXX <CR>' after prompt 'M:' causes execution to transfer to hex addr XXXX.

* 'S' SAVE to cassette *

Entering 'S <CR> after prompt 'M:' will invoke the cassette save sequence:

```
M: S <CR>
FILENAME: 'NAME OF PROGRAM'  <CR>
START:   1200  <CR>
END:     ZZZZ  <CR>
AUTO:    1200  <CR>
RECORD.PLAY
```

** EXPRESS PROGRAMMING HINTS **

1) The user has to make provision in memory for storing strings. it is suggested that the area above the compiler ($BB00 to $CFFF) be used for this purpose. A string is referred to not A$ but as A where A is the location at witch the string is stored. For exampie, the routines below accept an entry from the user, store it at hex $BB00 and then print it out.

```
10 N=$BB00:LINPUT N:PUSH N:CALL $1256:?/
'or'
10 LINPUT $BB00:PUSH $BB00:CALL $1256:?/
```

2) There is provision for single dimension arrays, eg: A(B)=C and there is no limit to array size. It is not necessary to DIM the array, but the user has instead to define the memory area where the storage of the array is to commence. Two bytes are used for each number. The exampie routine below establishes A(X) from A(0) to A(5) as having values of X*3 stores them from hex $BC00 on and prints them out.

```
10 A=$BC00:FOR X=0 TO 5:A(X)=X*3:? A(X),/:NEXT
```

3) Numeric arrays can be used to provide a string array facility. the numeric array variables being points to the memory location where the strings are stored. The routine below accepts 5 string entries from the user and later prints them out. The string themselves are stored at 41 byte intervals (LINPUT entries are padded out to 41 chr's) from hex $C500 on and the numeric arrays pointing to them at hex $C000 on.

```
10 K=$C000:FOR X=0 TO 4:K(X)=$C500+X*41:NEXT
20 FOR X=0 TO 4:?"ENTER ITEM ",X,/:LINPUT K(X):NEXT
30 FOR X=0 TO 4:PUSH K(X):CALL $1256:?/:NEXT
```

4) The interpreter normally cannot accept numbers greater than 32767, Although the limit does not apply to numbers used as PEEK or POKE etc... addresses, the interpreter will reject them if used in an associated FOR...NEXT loop, as eg: line 10 below. Employ the device used in the alternative line 20 to overcome this limitation.

```
10 FOR X=53248 TO 53258:POKE X,65:NEXT
20 FOR X=248 TO 258:POKE 53000+X.65:NEXT
```

5) Printer mode - correct protocol is 'PRMODE' not 'PR MODE' - the latter will cause syntax error.

6) When calling GET@ AND PUT@, the memory location for string storage may be specified in either hex or decimal; hex should be preceded by $ char,
eg: GET@ 0,0,39,34,50000 PUT@ 5,5,22,12,$C000

7) Programs can be written with EXPRESS without READ...DATA, the need for this only arising if programs intended for another interpreter are being adapted line by line.
Where the usage is to provide a machine code routine, this can be entered directly using EXPRESS monitor or written as BASIC and compiled. The other common application is to provide random access to strings, eg: for a vocabulary test as the first program below.
The second program demonstrates one way of simulating this with EXPRESS without the need to change the structure of the original.

```
10 W=5:REM W=No of word pairs +1
20 DATA "HOUSE","MAISON","DOG","CHIEN","DOOR","PORTE"
30 DATA "WINDOW","FENETRE"
40 RESTORE:FOR X=1 TO INT(RND(1)*W):READ Q$,A$:NEXT
50 PRINT "TRANSLATE ";Q$:INPUT G$
60 IF G$=A$ THEN PRINT "CORRECT":GOTO 40
70 PRINT "WRONG, THE ANSWER IS ";A$:GOTO 40
```

```
10 W=4:L=10:' W=No of word pairs, L=Max word length +2
20 Q=$C000:STA$(Q,"HOUSE   DOG     DOOR    WINDOW   ")
30 A=$C500:STA$(A,"MAISON  CHIEN   PORTE   FENETRE  ")
40 FOR X=1 TO W:POKE Q-1+X*L,13:POKE A-1+X*L,13:NEXT
50 V=(RND(W)-1)*L:PRINT "TRANSLATE ":PUSH Q+V:CALL $1256:?/
60 LINPUT $CA00:C=CP$(A+V,$CA00," "):IF C THEN PRINT"RIGHT"/:GOTO 50
70 PRINT"WRONG, THE ANSWER IS ":PUSH A+V:CALL $1256:?/:GOTO 50
```

NB: The function of line 40 is to add <CR> (ASCII 13) to the end of each subsidiary string.

** EXPRESS PROGRAMMING HINTS **

8) Negative STEP is not permitted; line 20 and 30 demonstrate two ways of implementing this to achieve the same effect as line 10.

```
10 FOR X=40 TO 10 STEP -1:PRINT X;:NEXT
20 X=40:FOR Y=10 TO 40:PRINT X:X=DEC(X):NEXT
30 X=40:REPEAT:PRINT X:X=DEC(X):UNTIL X=9
```

9) Although WOPEN and ROPEN are not provided, data files can be recorded and read back using the SHARP MONITOR routines CALL 33:CALL 36 and CALL 39:CALL 42 respectively, before writing data to cassette however.
It is first necessary to load $1102/03 with file length, $1104/05 with start address, $10F0 with 4 and if a file name is wanted, load $f10F1 onwards.

eg: Write a data block of 500 bytes from $C000 to $C4FF, filename 'AAA'

```
POKE $1102,$00:POKE $1103,$05:POKE $1104,$00:POKE $1105,$C0
POKE $10F1,65:POKE $10F2,65:POKE $10F3,65:POKE $10F4,13
POKE $10F0,4:CALL 33:CALL 36:?/
```

10) Full screen editing is provided. On the MZ80A editing can sometimes affect lower case/graphics char's within PRINT statements, necessitating entry of the complete section of the line within quotes.

11) Because of internal differences between the m/c's, code produced by MZ700 compiler will not necessarily run on the MZ80K, nor that from the MZ80k compiler on the MZ700 or MZ80A. This can be arranged however by POKING the values shown below into locations $18D6 and $1952 before saving the code. The cursor will be frozen on screen during the save routine, but can be restored to normal by POKING back the values shown in brackets on completion.

Using MZ700 compiler to run on MZ80K	$8F ($50)
Using MZ80K compiler to run on MZ80A & MZ700	$50 ($8F)

12) Because CP$ is not compilable, a routine has to be used for string comparisons if the program is to be compiled. Example 7 for instant could be re-written as below. If the comparison is to be frequently used, of course, it would be better to put the comparison in a sub-routine.

```
60 LINPUT $CA00:PRINT/
61 C=1:FOR R=0 TO L-1
62 IF(PEEK(A+V+R)=32)+(PEEK($CA00+R)=13) THEN R=L-1:GOTO 64
63 IF PEEK(A+V+R)<>PEEK($CA00+R) THEN C=0:R=L-1
64 NEXT:IF C THEN PRINT "RIGHT"/:GOTO 50
```

13) When compiling programs with very many variables, the start and end address may be scrolled off screen. This can be overcome by establishing PRMODE2 before calling the compiler, thus causing all data to be output to the printer. Alternatively, answer the 'M:' prompt at the end of compilation by '$DBA89 <CR>', to dump memory to screen. Press <Sp> to pause. $BA89/8A contains the start address for object code and $BAB3/B4 the address following end of object code.

** LOCATIONS **

$198C / $198D = BASIC TEXT START STORE ADDRESS
$198C / $198F = BASIC TEXT END STORE ADDRESS
$1850 = KEYBOARD LINE INPUT SUB ROUTINE, STORES AT $11A3
$18DC = CURSOR CHR ($EF)
$1695 = MONITOR ROUTINE
$1200 = COLD START
$1203 = WARM START
$1206 = MONITOR
$26a6 = BASIC FILE AUTO RUN FROM TAPE
$12A2 = PRINT MODE VALUE

```
PRINTER: LD   $12A2,1     ;PRINTMODE 1
         LD   HL,CHR      ;HL = POINTER TO ASC VALUE
         CALL $120C       ;PRINT ROUTINE
```

```
;*****************
;***           ***
;*** CMD ADDRs ***
;***           ***
;*****************

                ADDRES

2B0F 8F 26      DW 260FH        ;CMD RUN ADDR
2B11 67 20      DW 2067H        ;CMD LIST
2B13 3A 23      DW 233AH        ;CMD DELETE
2B15 E4 30      DW 30E4H        ;CMD VLIST
2B17 43 23      DW 2343H        ;CMD !
2B19 3B 21      DW 213BH        ;CMD NEW
2B1B 76 22      DW 2276H        ;CMD LOAD ADDR
2B1D DD 22      DW 22DDH        ;CMD SAVE ADDR
2B1F 00 F0      DW F000H        ;CMD DOS ADDR
2B21 3A 23      DW 233AH        ;CMD DIR
2B23 F3 23      DW 23F3H        ;CMD SYSTEM
2B25 6D 21      DW 216DH        ;CMD LIMLT
2B27 61 21      DW 2161H        ;CMD TEXT
2B29 43 21      DW 2143H        ;CMD CLR
2B2B D4 20      DW 20D4H        ;CMD LDELETE
2B2D 3A 23      DW 323AH        ;CMD LOCK
2B2F 21 27      DW 2721H        ;CMD IF
2B31 72 27      DW 2772H        ;CMD THEN
2B33 BE 27      DW 27BEH        ;CMD FOR
2B35 28 28      DW 2828H        ;CMD NEXT
2B37 73 27      DW 2773H        ;CMD REPEAT
2B39 92 27      DW 2792H        ;CMD UNTIL
2B3B 98 28      DW 2898H        ;CMD ON
2B3D 1B 27      DW 271BH        ;CMD GOTO
2B3F 2E 27      DW 272EH        ;CMD GOSUB
2B41 4E 27      DW 274EH        ;CMD RETURN
2B43 27 27      DW 2727H        ;CMD REM
2B45 74 21      DW 2174H        ;CMD ?
2B47 2E 22      DW 222EH        ;CMD INPUT
2B49 3A 23      DW 233AH        ;CMD RENAME
2B4B E9 26      DW 26E9H        ;CMD STOP
2B4D 95 28      DW 2895H        ;CMD END
2B4F 95 16      DW 1695H        ;CMD MON
2B51 00 00      DW 0000H        ;CMD BYE
2B53 30 21      DW 2130H        ;CMD CALL
2B55 E2 28      DW 28E2H        ;CMD PUSH
2B57 FF 28      DW 28FFH        ;CMD POP
2B59 D1 2C      DW 2CD1H        ;CMD OUT
2B5B 0E 22      DW 220EH        ;CMD CUR@
2B5D 4D 24      DW 244DH        ;CMD POKE
2B5F 9F 30      DW 309FH        ;CMD STA$(
2B61 75 2C      DW 2C75H        ;CMD LINPUT

2B63 27 27      DW 2727H        ;CMD '          :REM CMD

2B65 3A 23      DW 233AH        ;CMD UNLOCK
2B67 07 23      DW 2307H        ;CMD APPEND
2B69 00 98      DW 9800H        ;CMD COMP

2B6B 63 24      DW 2463H        ;CMD TO
2B6D 63 24      DW 2463H        ;CMD STEP

2B6F 7C 22      DW 227CH        ;CMD EXCH

2B71 BC 2C      DW 2CBCH        ;CMD SPOKE
2B73 21 22      DW 2221H        ;CMD CURH
2B75 14 22      DW 2214H        ;CMD CURV
2B77 F7 2B      DW 2BF7H        ;CMD PLOT
2B79 0F 2C      DW 2C0FH        ;CMD DRAW
2B7B 54 2C      DW 2C54H        ;CMD PUT@
2B7D 33 2C      DW 2C33H        ;CMD GET@
2B7F 58 24      DW 2458H        ;CMD PRMODE
2B81 3E 00      DW 003EH        ;CMD BELL
2B83 94 2C      DW 2C94H        ;CMD MUSIC
2B85 88 2C      DW 2C88H        ;CMD TEMPO
2B87 7E 2C      DW 2C7EH        ;CMD SOUND
2B89 D9 30      DW 30D9H        ;CMD TIME=
2B8B 63 24      DW 2463H        ;CMD xxxx
2B8D 48 23      DW 2348H        ;CMD #S
2B8F 81 23      DW 2381H        ;CMD #L

2B91 63 24      DW 2463H        ;CMD
2B93 63 24      DW 2463H        ;CMD
2B95 63 24      DW 2463H        ;CMD
2B97 63 24      DW 2463H        ;CMD
2B99 63 24      DW 2463H        ;CMD
2B9B 63 24      DW 2463H        ;CMD
2B9D 63 24      DW 2463H        ;CMD

2B9F ED 2F      DW 2FEDH        ;CMD SPEEK(
2BA1 E0 30      DW 30E0H        ;CMD TIME

2BA3 63 24      DW 2463H        ;CMD
2BA5 63 24      DW 2463H        ;CMD
2BA7 63 24      DW 2463H        ;CMD

2BA9 63 30      DW 3063H        ;CMD SEARCH$(
2BAB 4E 30      DW 304EH        ;CMD CP$(
2BAD 72 30      DW 3072H        ;CMD CPI(
2BAF B6 14      DW 14B6H        ;CMD GET
2BB1 B9 14      DW 14B9H        ;CMD INKEY
2BB3 1A 30      DW 301AH        ;CMD MOD(
2BB5 05 30      DW 3005H        ;CMD ABS(
2BB7 0B 30      DW 300BH        ;CMD SGN(
2BB9 11 30      DW 3011H        ;CMD RND(
2BBB FC 2F      DW 2FFCH        ;CMD INP(
2BBD C1 30      DW 30C1H        ;CMD ADRS(
2BBF E6 2F      DW 2FE6H        ;CMD PEEK(
2BC1 CB 2F      DW 2FCBH        ;CMD INC(
2BC3 DC 2F      DW 2FDCH        ;CMD DEC(
2BC5 C8 30      DW 30C8H        ;CMD USR(

2BC7 63 24      DW 2463H        ;CMD
2BC9 63 24      DW 2463H        ;CMD
2BCB 63 24      DW 2463H        ;CMD
2BCD 63 24      DW 2463H        ;CMD

2BCF 63 24      DW 2463H        ;CMD <=
2BD1 63 24      DW 2463H        ;CMD =<
2BD3 63 24      DW 2463H        ;CMD >=
2BD5 63 24      DW 2463H        ;CMD =>
2BD7 63 24      DW 2463H        ;CMD =
2BD9 63 24      DW 2463H        ;CMD <>
2BDB 63 24      DW 2463H        ;CMD <
2BDD 63 24      DW 2463H        ;CMD >
2BDF 63 24      DW 2463H        ;CMD OR
2BE1 63 24      DW 2463H        ;CMD +
2BE3 63 24      DW 2463H        ;CMD -
2BE5 63 24      DW 2463H        ;CMD *
2BE7 63 24      DW 2463H        ;CMD /
2BE9 63 24      DW 2463H        ;CMD AND
2BEB 63 24      DW 2463H        ;CMD XOR
2BED 63 24      DW 2463H        ;CMD
2BEF 63 24      DW 2463H        ;CMD
2BF1 63 24      DW 2463H        ;CMD
2BF3 63 24      DW 2463H        ;CMD
2BF5 63 24      DW 2463H        ;CMD
```

```
;*****************
;***           ***
;*** CMD LIST ***
;***           ***
;*****************

                ASCII

291F 3A 52 55 4E                DB      ':RUN'
2923 3A 4C 49 53 54             DB      ':LIST'
2928 3A 44 45 4C 45 54 45       DB      ':DELETE'
292F 3A 56 4C 49 53 54          DB      ':VLIST'
2935 3A 21                      DB      ':!'
2937 3A 4E 45 57                DB      ':NEW'
293B 3A 4C 4F 41 44             DB      ':LOAD'
2940 3A 53 41 56 45             DB      ':SAVE'
2945 3A 44 4F 53                DB      ':DOS'
2949 3A 44 49 52                DB      ':DIR'
294D 3A 53 59 53 54 45 4D       DB      ':SYSTEM'
2954 3A 4C 49 4D 49 54          DB      ':LIMIT'
295A 3A 54 45 58 54             DB      ':TEXT'
295F 3A 43 4C 52                DB      ':CLR'
2963 3A 4C 44 45 4C 45 54 45    DB      ':LDELETE'
296B 3A 4C 4F 43 4B             DB      ':LOCK'
2970 3A 49 46                   DB      ':IF'
2973 3A 54 48 45 4E             DB      ':THEN'
2978 3A 46 4F 52                DB      ':FOR'
297C 3A 4E 45 58 54             DB      ':NEXT'
2981 3A 52 45 50 45 41 54       DB      ':REPEAT'
2988 3A 55 4E 54 49 4C          DB      ':UNTIL'
298E 3A 4F 4E                   DB      ':ON'
2991 3A 47 4F 54 4F             DB      ':GOTO'
2996 3A 47 4F 53 55 42          DB      ':GOSUB'
299C 3A 52 45 54 55 52 4E       DB      ':RETURN'
29A3 3A 52 45 4D 20             DB      ':REM '
29A8 3A 3F                      DB      ':?'
29AA 3A 49 4E 50 55 54          DB      ':INPUT'
29B0 3A 52 45 5E 41 4D 45       DB      ':RENAME'
29B7 3A 53 54 4F 50             DB      ':STOP'
29BC 3A 45 4E 44                DB      ':END'
29C0 3A 4D 4F 4E                DB      ':MON'
29C4 3A 42 59 45                DB      ':BYE'
29C8 3A 43 41 4C 4C             DB      ':CALL'
29CD 3A 50 55 53 48             DB      ':PUSH'
29D2 3A 50 4F 50                DB      ':POP'
29D6 3A 4F 55 54                DB      ':OUT'
29DA 3A 43 55 52 4D             DB      ':CUR@'
29DF 3A 50 4F 4B 45             DB      ':POKE'
29E4 3A 53 54 41 24 28          DB      ':STA$('
29EA 3A 4C 49 4E 50 55 54       DB      ':LINPUT'

29F1 3A 27 3A 55 4E 4C 4F 43 4B DB      ':UNLOCK'
29FA 3A 41 50 50 45 4E 44       DB      ':APPEND'
2A01 3A 43 4F 4D 50             DB      ':COMP'

2A06 3A 54 4F                   DB      ':TO'
2A09 3A 53 54 45 50             DB      ':STEP'

2A0E 3A 45 58 43 48             DB      ':EXCH'

2A13 3A 53 50 4F 4B 45          DB      ':SPOKE'
2A19 3A 43 55 52 48             DB      ':CURH'
2A1E 3A 43 55 52 56             DB      ':CURV'
2A23 3A 50 4C 4F 54             DB      ':PLOT'
2A28 3A 44 52 41 57             DB      ':DRAW'
2A2D 3A 50 55 54 40             DB      ':PUT@'
2A32 3A 47 45 54 40             DB      ':GET@'
2A37 3A 50 52 4D 4F 45          DB      ':PRMODE'
2A3E 3A 42 45 4C 4C             DB      ':BELL'
2A43 3A 4D 55 53 49 43          DB      ':MUSIC'
2A49 3A 54 45 4D 50 4F          DB      ':TEMPO'
2A4F 3A 53 4F 55 4E 44          DB      ':SOUND'
2A55 3A 54 49 4D 45 3D          DB      ':TIME='
2A5B 3A 63 63 63 63             DB      ':XXXX
2A60 3A 23 53                   DB      ':#S'
2A63 3A 23 4C                   DB      ':#L

2A66 3A 01                      DB
2A68 3A 01                      DB
2A6A 3A 01                      DB
2A6C 3A 01                      DB
2A6E 3A 01                      DB
2A70 3A 01                      DB
2A72 3A 01                      DB

2A74 3A 53 50 45 45 4B 28       DB      ':SPEEK('
2A7B 3A 54 49 4D 45             DB      ':TIME'

2A80 3A 01                      DB
2A82 3A 01                      DB
2A84 3A 01                      DB

2A86 3A 53 45 41 52 43 48 24 28 DB      ':SEARCH$('
2A8F 3A 43 50 24 28             DB      ':CP$('
2A94 3A 43 50 49 28             DB      ':CPI('
2A99 3A 47 45 54                DB      ':GET'
2A9D 3A 49 4E 4B 45 59          DB      ':INKEY'
2AA3 3A 4D 4F 44 28             DB      ':MOD('
2AA8 3A 41 42 53 28             DB      ':ABS('
2AAD 3A 53 47 4E 28             DB      ':SGN('
2AB2 3A 52 4E 44 28             DB      ':RND('
2AB7 3A 49 4E 50 28             DB      ':INP('
2ABC 3A 41 44 52 53 28          DB      ':ADRS('
2AC2 3A 50 45 45 4B 28          DB      ':PEEK('
2AC8 3A 49 4E 43 28             DB      ':INC('
2ACD 3A 44 45 43 28             DB      ':DEC('
2AD2 3A 55 53 52 28             DB      ':USER('

2AD7 3A 01                      DB
2AD9 3A 01                      DB
2ADB 3A 01                      DB
2ADD 3A 01                      DB

2ADF 3A 3C 3D                   DB      ':<='
2AE2 3A 3D 3C                   DB
2AE5 3A 3E 3D                   DB
2AE8 3A 3D 3E                   DB
2AEB 3A 3D                      DB
2AED 3A 3C 3E                   DB
2AF0 3A 3C                      DB
2AF2 3A 3E                      DB
2AF4 3A 20 4F 52 20             DB
2AF9 3A 2B                      DB
2AFB 3A 2D                      DB
2AFD 3A 2A                      DB      ':*'
2AFF 3A 2F                      DB      ':/'            CRLF
2B01 3A 20 41 4E 44 20          DB      ': AND '
2B07 3A 20 58 4F 52 20          DB      ': XOR '
2B0D 3A 00                      DB      ':'
```

MESSAGES TXT

$259D = "READY"
$2467 = "SYNTAK"
$2472 = "ERROR"
$26ED = "BREAK"
$26FF = "IN"
$167C = "DATA ER."
$24AF = "ILL FUNCTION CALL"
$24C7 = "MOMORY"
$24D4 = "UNDER LINE NO:"
$24E9 = "STACK EMPTY"
$24FC = "BAD NEXT"
$250C = "BAD RETURN"
$251E = "BAD UNTIL"
$252F = "STACK OVER"

$2543 = "DISK EXPRESS BASIC V1.04"

$168C = "DIV ER."
$17C0 = "BREAK"

$1A0C = "*PRINTER ERR"
$23F6 = " SYSTEM **"**
$2408 = "TEXT START:"
$2420 = "TEXT END :"
$2438 = "END MEMORY:"

ADDR:
$198C = LOW BYTE TEXT START
$198D = HIGH BYTE TEXT START
$198E = LOW BYTE TEXT END
$198F = HIGH BYTE TEXT END

```
1200 C3 DB 13    L0585:  JP L0001        ;COLD START
1203 C3 94 25            JP L0002        ;WARMSTART      'JUMP FROM MON'
1206 C3 95 16    L0906:  JP L0003        ;MON
1209 C3 E9 13    L0586:  JP <CR>         ;CRLF TWICE
120C C3 1E 14    L0244:  JP L0005        ;? & CHR
120F C3 21 14    L0874:  JP ?DEC         ;? DECIMAL (HL POINTS TO VARIABLE)
1212 C3 26 14    L0862:  JP L0007
1215 C3 4C 14    L0595:  JP L0008        ;? # FUNCTION
1218 C3 4F 14            JP L0009
121B C3 53 14            JP L0010
121E C3 5D 14    L0587:  JP L0011        ;? MSG, END=00 + CTRL CODES
1221 C3 68 14            JP L0012
1224 C3 B6 14            JP L0013        ;ALTERNATE CURSOR & CHR ON SCREEN
1227 C3 B9 14            JP L0014        ;INKEY (GET KEY FROM KB NO WAIT: Acc=0 NO KEY!)
122A C3 CE 14            JP L0015
122D C3 00 00            JP L0016
1230 C3 00 00            JP L0016
1233 C3 21 15            JP L0017
1236 C3 C1 15            JP L0018        ;GET@ CMD
1239 C3 0F 16            JP L0019        ;PUT@ CMD
123C C3 3C 18            JP L0020        ;SPOKE X,Y,CHR
123F C3 29 18            JP L0021        ;SPEEK
1242 C3 16 18            JP L0022        ;MON READ TIMEaqz\
1245 C3 1B 18            JP L0023        ;TIME = 0
1248 C3 02 18            JP MATCH        ;CHR MATCH
124B C3 F4 17            JP L0025
124E C3 21 18            JP L0026        ;copy chrs ending with 0 (DE=SORCE, HL=DEST)
1251 C3 D1 17            JP L0027

;** PRINT TO SCREEN CHRS UNTIL $0D **

1256 CD 1F 13            CALL L0028     ;USER CALL ROUTINE FROM BASIC
1259 22 54 12            LD (1254H),HL  ;EXCH HL,DE
125C ED 5B 54 12         LD DE,(1254H)
1260 CD 15 00            CALL L0029     ;MON ? MSG
1263 C9                  RET

1264 00                  .DB 0
1265 00                  .DB 0
```

ASCII Code Chars:

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
0		32	SP	64	@	96	♠	128	SP	160	q	192	SP	224	
1		33	!	65	A	97	H	129		161	a	193		225	
2		34	"	66	B	98		130		162	z	194		226	
3		35	#	67	C	99		131		163	w	195		227	
4		36	$	68	D	100		132		164	s	196		228	
5		37	%	69	E	101		133		165	u	197		229	
6		38	&	70	F	102	¥	134		166	i	198		230	
7		39	'	71	G	103	☺	135		167		199		231	
8		40	(	72	H	104	☺	136		168	ö	200		232	
9		41	)	73	I	105		137		169	k	201		233	
10		42	*	74	J	106		138		170	f	202		234	
11		43	+	75	K	107		139		171	v	203		235	
12		44	,	76	L	108	K	140		172		204		236	
13	CR	45	-	77	M	109	K	141		173	ü	205		237	
14		46	.	78	N	110		142		174	ß	206		238	
15		47	/	79	O	111		143		175	j	207		239	
16		48	0	80	P	112		144		176	n	208		240	
17	↓	49	1	81	Q	113		145		177	/	209		241	
18	↑	50	2	82	R	114		146	e	178	ü	210		242	
19	→	51	3	83	S	115		147		179	m	211		243	
20	←	52	4	84	T	116		148		180		212		244	
21	H	53	5	85	U	117		149		181		213		245	
22	C	54	6	86	V	118		150	t	182		214		246	
23		55	7	87	W	119		151	g	183	o	215		247	
24		56	8	88	X	120		152	h	184	l	216		248	
25		57	9	89	Y	121		153		185	Ä	217		249	
26		58	:	90	Z	122		154	b	186	ö	218		250	
27		59	;	91	[	123		155	x	187	ä	219		251	£
28		60	<	92	\	124		156	d	188		220		252	
29		61	=	93	]	125		157	r	189	y	221		253	
30		62	>	94	↑	126		158	p	190		222		254	
31		63	?	95	←	127		159	c	191		223		255	

Display Code Chars:

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
0	SP	32	0	64	SP	96	7/	128	SP	160		192		224	
1	A	33	1	65		97	!	129	a	161		193		225	
2	B	34	2	66		98	"	130	b	162		194		226	
3	C	35	3	67		99	#	131	c	163		195		227	
4	D	36	4	68		100	$	132	d	164		196		228	
5	E	37	5	69		101	%	133	e	165		197		229	
6	F	38	6	70		102	&	134	f	166		198		230	
7	G	39	7	71		103	'	135	g	167		199		231	
8	H	40	8	72		104	(	136	h	168		200		232	
9	I	41	9	73	?	105	)	137	i	169		201		233	
10	J	42	-	74		106	+	138	j	170	ß	202		234	
11	K	43	=	75		107		139	k	171	ü	203		235	
12	L	44	;	76		108		140	l	172	ö	204		236	
13	M	45	/	77		109		141	m	173	ü	205		237	
14	N	46	.	78		110		142	n	174	Ä	206		238	
15	O	47	,	79	:	111		143	o	175	Ö	207		239	
16	P	48		80		112		144	p	176		208		240	SP
17	Q	49		81		113		145	q	177		209		241	
18	R	50		82		114		146	r	178		210		242	
19	S	51		83		115		147	s	179		211		243	
20	T	52		84		116		148	t	180		212		244	
21	U	53		85	@	117		149	u	181		213		245	
22	V	54		86		118		150	v	182		214		246	
23	W	55		87		119		151	w	183		215		247	
24	X	56		88		120		152	x	184		216		248	
25	Y	57		89		121		153	y	185		217		249	
26	Z	58		90		122		154	z	186		218		250	
27	£	59		91		123		155	ä	187		219		251	
28		60		92		124		156		188		220		252	
29		61		93		125		157		189		221		253	
30		62		94		126		158		190		222		254	
31		63		95		127		159		191		223		255	

ASCII Code Chars:

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00		32	20	SP	64	40	@	96	60	`	128	80	SP	160	A0	q	192	C0	SP	224	E0	
1	01		33	21	!	65	41	A	97	61	a	129	81		161	A1	a	193	C1		225	E1	
2	02		34	22	"	66	42	B	98	62	b	130	82		162	A2	z	194	C2		226	E2	
3	03		35	23	#	67	43	C	99	63	c	131	83		163	A3	w	195	C3		227	E3	
4	04		36	24	$	68	44	D	100	64	d	132	84		164	A4	s	196	C4		228	E4	
5	05		37	25	%	69	45	E	101	65	e	133	85		165	A5	u	197	C5		229	E5	
6	06		38	26	&	70	46	F	102	66	f	134	86		166	A6	i	198	C6		230	E6	
7	07		39	27	'	71	47	G	103	67	g	135	87		167	A7		199	C7		231	E7	
8	08		40	28	(	72	48	H	104	68	h	136	88		168	A8	ö	200	C8		232	E8	
9	09		41	29	)	73	49	I	105	69	i	137	89		169	A9	k	201	C9		233	E9	
10	0A		42	2A	*	74	4A	J	106	6A	j	138	8A		170	AA	f	202	CA		234	EA	
11	0B		43	2B	+	75	4B	K	107	6B	k	139	8B		171	AB	v	203	CB		235	EB	
12	0C		44	2C	,	76	4C	L	108	6C	l	140	8C		172	AC		204	CC		236	EC	
13	0D	CR	45	2D	-	77	4D	M	109	6D	m	141	8D		173	AD	ü	205	CD		237	ED	
14	0E		46	2E	.	78	4E	N	110	6E	n	142	8E		174	AE	ß	206	CE		238	EE	
15	0F		47	2F	/	79	4F	O	111	6F	o	143	8F		175	AF	j	207	CF		239	EF	
16	10		48	30	0	80	50	P	112	70	p	144	90		176	B0	n	208	D0		240	F0	
17	11	↓	49	31	1	81	51	Q	113	71	q	145	91		177	B1	/	209	D1		241	F1	
18	12	↑	50	32	2	82	52	R	114	72	r	146	92	e	178	B2	ü	210	D2		242	F2	
19	13	→	51	33	3	83	53	S	115	73	s	147	93		179	B3	m	211	D3		243	F3	
20	14	←	52	34	4	84	54	T	116	74	t	148	94		180	B4		212	D4		244	F4	
21	15	H	53	35	5	85	55	U	117	75	u	149	95		181	B5		213	D5		245	F5	
22	16	C	54	36	6	86	56	V	118	76	v	150	96	t	182	B6		214	D6		246	F6	
23	17		55	37	7	87	57	W	119	77	w	151	97	g	183	B7	o	215	D7		247	F7	
24	18		56	38	8	88	58	X	120	78	x	152	98	h	184	B8		216	D8		248	F8	
25	19		57	39	9	89	59	Y	121	79	y	153	99		185	B9	Ä	217	D9		249	F9	
26	1A		58	3A	:	90	5A	Z	122	7A	z	154	9A	b	186	BA	ö	218	DA		250	FA	
27	1B		59	3B	;	91	5B	[	123	7B	{	155	9B	x	187	BB	ä	219	DB		251	FB	£
28	1C		60	3C	<	92	5C	\	124	7C			156	9C	d	188	BC		220	DC		252	FC	
29	1D		61	3D	=	93	5D	]	125	7D	}	157	9D	r	189	BD	y	221	DD		253	FD	
30	1E		62	3E	>	94	5E	↑	126	7E	~	158	9E	p	190	BE		222	DE		254	FE	
31	1F		63	3F	?	95	5F	←	127	7F		159	9F	c	191	BF		223	DF		255	FF	ƒ

Display Code Chars:

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	SP	32	20	0	64	40	SP	96	60	`	128	80	SP	160	A0		192	C0		224	E0	
1	01	A	33	21	1	65	41		97	61	!	129	81	a	161	A1		193	C1		225	E1	
2	02	B	34	22	2	66	42		98	62	"	130	82	b	162	A2		194	C2		226	E2	
3	03	C	35	23	3	67	43		99	63	#	131	83	c	163	A3		195	C3		227	E3	
4	04	D	36	24	4	68	44		100	64	$	132	84	d	164	A4		196	C4		228	E4	
5	05	E	37	25	5	69	45		101	65	%	133	85	e	165	A5		197	C5		229	E5	
6	06	F	38	26	6	70	46		102	66	&	134	86	f	166	A6		198	C6		230	E6	
7	07	G	39	27	7	71	47		103	67	'	135	87	g	167	A7		199	C7		231	E7	
8	08	H	40	28	8	72	48		104	68	(	136	88	h	168	A8		200	C8		232	E8	
9	09	I	41	29	9	73	49	?	105	69	)	137	89	i	169	A9		201	C9		233	E9	
10	0A	J	42	2A	-	74	4A		106	6A	+	138	8A	j	170	AA	ß	202	CA		234	EA	
11	0B	K	43	2B	=	75	4B		107	6B	*	139	8B	k	171	AB	ü	203	CB		235	EB	
12	0C	L	44	2C	;	76	4C		108	6C		140	8C	l	172	AC	ö	204	CC		236	EC	
13	0D	M	45	2D	/	77	4D		109	6D		141	8D	m	173	AD	ü	205	CD		237	ED	
14	0E	N	46	2E	.	78	4E		110	6E		142	8E	n	174	AE	Ä	206	CE		238	EE	
15	0F	O	47	2F	,	79	4F		111	6F		143	8F	o	175	AF	ö	207	CF		239	EF	
16	10	P	48	30		80	50		112	70		144	90	p	176	B0		208	D0		240	F0	SP
17	11	Q	49	31		81	51		113	71		145	91	q	177	B1		209	D1		241	F1	
18	12	R	50	32		82	52		114	72		146	92	r	178	B2		210	D2		242	F2	
19	13	S	51	33		83	53		115	73		147	93	s	179	B3		211	D3		243	F3	
20	14	T	52	34		84	54		116	74		148	94	t	180	B4		212	D4		244	F4	
21	15	U	53	35		85	55	@	117	75		149	95	u	181	B5		213	D5		245	F5	
22	16	V	54	36		86	56		118	76		150	96	v	182	B6		214	D6		246	F6	
23	17	W	55	37		87	57		119	77		151	97	w	183	B7		215	D7		247	F7	
24	18	X	56	38		88	58		120	78		152	98	x	184	B8		216	D8		248	F8	
25	19	Y	57	39		89	59		121	79		153	99	y	185	B9		217	D9		249	F9	
26	1A	Z	58	3A		90	5A		122	7A		154	9A	z	186	BA		218	DA		250	FA	
27	1B	£	59	3B		91	5B		123	7B		155	9B	ä	187	BB		219	DB		251	FB	
28	1C		60	3C		92	5C		124	7C		156	9C		188	BC		220	DC		252	FC	
29	1D		61	3D		93	5D		125	7D		157	9D		189	BD		221	DD		253	FD	
30	1E		62	3E		94	5E		126	7E		158	9E		190	BE		222	DE		254	FE	
31	1F		63	3F		95	5F		127	7F		159	9F		191	BF		223	DF		255	FF	

*256			*256			*256			*256			*256		
HEX	DEC	DEC	HEX	DEC	DEC	HEX	DEC	DEC	HEX	DEC	DEC	HEX	DEC	DEC
00	00000	0	34	13312	52	68	26624	104	9C	39936	156	D0	53248	208
01	00256	1	35	13568	53	69	26880	105	9D	40192	157	D1	53504	209
02	00512	2	36	13824	54	6A	27136	106	9E	40448	158	D2	53760	210
03	00768	3	37	14080	55	6B	27392	107	9F	40704	159	D3	54016	211
04	01024	4	38	14336	56	6C	27648	108	A0	40960	160	D4	54272	212
05	01280	5	39	14592	57	6D	27904	109	A1	41216	161	D5	54528	213
06	01536	6	3A	14848	58	6E	28160	110	A2	41472	162	D6	54784	214
07	01792	7	3B	15104	59	6F	28416	111	A3	41728	163	D7	55040	215
08	02048	8	3C	15360	60	70	28672	112	A4	41984	164	D8	55296	216
09	02304	9	3D	15616	61	71	28928	113	A5	42240	165	D9	55552	217
0A	02560	10	3E	15872	62	72	29184	114	A6	42496	166	DA	55808	218
0B	02816	11	3F	16128	63	73	29440	115	A7	42752	167	DB	56064	219
0C	03072	12	40	16384	64	74	29696	116	A8	43008	168	DC	56320	220
0D	03328	13	41	16640	65	75	29952	117	A9	43264	169	DD	56576	221
0E	03584	14	42	16896	66	76	30208	118	AA	43520	170	DE	56832	222
0F	03840	15	43	17152	67	77	30464	119	AB	43776	171	DF	57088	223
10	04096	16	44	17408	68	78	30720	120	AC	44032	172	E0	57344	224
11	04352	17	45	17664	69	79	30976	121	AD	44288	173	E1	57600	225
12	04608	18	46	17920	70	7A	31232	122	AE	44544	174	E2	57956	226
13	04864	19	47	18176	71	7B	31488	123	AF	44800	175	E3	58112	227
14	05120	20	48	18432	72	7C	31744	124	B0	45056	176	E4	58368	228
15	05376	21	49	18688	73	7D	32000	125	B1	45312	177	E5	58624	229
16	05632	22	4A	18944	74	7E	32256	126	B2	45568	178	E6	58880	230
17	05888	23	4B	19200	75	7F	32512	127	B3	45824	179	E7	59136	231
18	06144	24	4C	19456	76	80	32768	128	B4	46080	180	E8	59392	232
19	06400	25	4D	19712	77	81	33024	129	B5	46336	181	E9	59648	233
1A	06656	26	4E	19968	78	82	33280	130	B6	46592	182	EA	59904	234
1B	06912	27	4F	20224	79	83	33536	131	B7	46848	183	EB	60160	235
1C	07166	28	50	20480	80	84	33792	132	B8	47104	184	EC	60416	236
1D	07424	29	51	20736	81	85	34048	133	B9	47360	185	ED	60672	237
1E	07680	30	52	20992	82	86	34304	134	BA	47616	186	EE	60928	238
1F	07936	31	53	21248	83	87	34560	135	BB	47872	187	EF	61184	239
20	08192	32	54	21504	84	88	34816	136	BC	48128	188	F0	61440	240
21	08448	33	55	21760	85	89	35072	137	BD	48384	189	F1	61696	241
22	08704	34	56	22016	86	8A	35328	138	BE	48640	190	F2	61952	242
23	08960	35	57	22272	87	8B	35584	139	BF	48896	191	F3	62208	243
24	09216	36	58	22528	88	8C	35840	140	C0	49152	192	F4	62464	244
25	09472	37	59	22784	89	8D	36096	141	C1	49408	193	F5	62720	245
26	09728	38	5A	23040	90	8E	36352	142	C2	49664	194	F6	62976	246
27	09984	39	5B	23296	91	8F	36608	143	C3	49920	195	F7	63232	247
28	10240	40	5C	23552	92	90	36864	144	C4	50176	196	F8	63488	248
29	10496	41	5D	23808	93	91	37120	145	C5	50432	197	F9	63744	249
2A	10752	42	5E	24064	94	92	37376	146	C6	50688	198	FA	64000	250
2B	11008	43	5F	24320	95	93	37632	147	C7	50944	199	FB	64256	251
2C	11264	44	60	24576	96	94	37888	148	C8	51200	200	FC	64512	252
2D	11520	45	61	24832	97	95	38144	149	C9	51456	201	FD	64678	253
2E	11776	46	62	25088	98	96	38400	150	CA	51712	202	FE	65024	254
2F	12032	47	63	25344	99	97	38656	151	CB	51968	203	FF	65280	255
30	12288	48	64	25600	100	98	38912	152	CC	52224	204			
31	12544	49	65	25856	101	99	39168	153	CD	52480	205			
32	12800	50	66	26112	102	9A	39424	154	CE	52736	206			
33	13056	51	67	26368	103	9B	39680	155	CF	52992	207			

In each row the first column is the HEX code.
The second column is the DECIMAL equivalent multiplied by 256 for calculating the M.S.B.
The third column is for use with the L.S.B.

DEC	HEX	BINARY	DEC	HEX	BINARY	DEC	HEX	BINARY	DEC	HEX	BINARY
0	00	00000000	64	40	01000000	128	80	10000000	192	C0	11000000
1	01	00000001	65	41	01000001	129	81	10000001	193	C1	11000001
2	02	00000010	66	42	01000010	130	82	10000010	194	C2	11000010
3	03	00000011	67	43	01000011	131	83	10000011	195	C3	11000011
4	04	00000100	68	44	01000100	132	84	10000100	196	C4	11000100
5	05	00000101	69	45	01000101	133	85	10000101	197	C5	11000101
6	06	00000110	70	46	01000110	134	86	10000110	198	C6	11000110
7	07	00000111	71	47	01000111	135	87	10000111	199	C7	11000111
8	08	00001000	72	48	01001000	136	88	10001000	200	C8	11001000
9	09	00001001	73	49	01001001	137	89	10001001	201	C9	11001001
10	0A	00001010	74	4A	01001010	138	8A	10001010	202	CA	11001010
11	0B	00001011	75	4B	01001011	139	8B	10001011	203	CB	11001011
12	0C	00001100	76	4C	01001100	140	8C	10001100	204	CC	11001100
13	0D	00001101	77	4D	01001101	141	8D	10001101	205	CD	11001101
14	0E	00001110	78	4E	01001110	142	8E	10001110	206	CE	11001110
15	0F	00001111	79	4F	01001111	143	8F	10001111	207	CF	11001111
16	10	00010000	80	50	01010000	144	90	10010000	208	D0	11010000
17	11	00010001	81	51	01010001	145	91	10010001	209	D1	11010001
18	12	00010010	82	52	01010010	146	92	10010010	210	D2	11010010
19	13	00010011	83	53	01010011	147	93	10010011	211	D3	11010011
20	14	00010100	84	54	01010100	148	94	10010100	212	D4	11010100
21	15	00010101	85	55	01010101	149	95	10010101	213	D5	11010101
22	16	00010110	86	56	01010110	150	96	10010110	214	D6	11010110
23	17	00010111	87	57	01010111	151	97	10010111	215	D7	11010111
24	18	00011000	88	58	01011000	152	98	10011000	216	D8	11011000
25	19	00011001	89	59	01011001	153	99	10011001	217	D9	11011001
26	1A	00011010	90	5A	01011010	154	9A	10011010	218	DA	11011010
27	1B	00011011	91	5B	01011011	155	9B	10011011	219	DB	11011011
28	1C	00011100	92	5C	01011100	156	9C	10011100	220	DC	11011100
29	1D	00011101	93	5D	01011101	157	9D	10011101	221	DD	11011101
30	1E	00011110	94	5E	01011110	158	9E	10011110	222	DE	11011110
31	1F	00011111	95	5F	01011111	159	9F	10011111	223	DF	11011111
32	20	00100000	96	60	01100000	160	A0	10100000	224	E0	11100000
33	21	00100001	97	61	01100001	161	A1	10100001	225	E1	11100001
34	22	00100010	98	62	01100010	162	A2	10100010	226	E2	11100010
35	23	00100011	99	63	01100011	163	A3	10100011	227	E3	11100011
36	24	00100100	100	64	01100100	164	A4	10100100	228	E4	11100100
37	25	00100101	101	65	01100101	165	A5	10100101	229	E5	11100101
38	26	00100110	102	66	01100110	166	A6	10100110	230	E6	11100110
39	27	00100111	103	67	01100111	167	A7	10100111	231	E7	11100111
40	28	00101000	104	68	01101000	168	A8	10101000	232	E8	11101000
41	29	00101001	105	69	01101001	169	A9	10101001	233	E9	11101001
42	2A	00101010	106	6A	01101010	170	AA	10101010	234	EA	11101010
43	2B	00101011	107	6B	01101011	171	AB	10101011	235	EB	11101011
44	2C	00101100	108	6C	01101100	172	AC	10101100	236	EC	11101100
45	2D	00101101	109	6D	01101101	173	AD	10101101	237	ED	11101101
46	2E	00101110	110	6E	01101110	174	AE	10101110	238	EE	11101110
47	2F	00101111	111	6F	01101111	175	AF	10101111	239	EF	11101111
48	30	00110000	112	70	01110000	176	B0	10110000	240	F0	11110000
49	31	00110001	113	71	01110001	177	B1	10110001	241	F1	11110001
50	32	00110010	114	72	01110010	178	B2	10110010	242	F2	11110010
51	33	00110011	115	73	01110011	179	B3	10110011	243	F3	11110011
52	34	00110100	116	74	01110100	180	B4	10110100	244	F4	11110100
53	35	00110101	117	75	01110101	181	B5	10110101	245	F5	11110101
54	36	00110110	118	76	01110110	182	B6	10110110	246	F6	11110110
55	37	00110111	119	77	01110111	183	B7	10110111	247	F7	11110111
56	38	00111000	120	78	01111000	184	B8	10111000	248	F8	11111000
57	39	00111001	121	79	01111001	185	B9	10111001	249	F9	11111001
58	3A	00111010	122	7A	01111010	186	BA	10111010	250	FA	11111010
59	3B	00111011	123	7B	01111011	187	BB	10111011	251	FB	11111011
60	3C	00111100	124	7C	01111100	188	BC	10111100	252	FC	11111100
61	3D	00111101	125	7D	01111101	189	BD	10111101	253	FD	11111101
62	3E	00111110	126	7E	01111110	190	BE	10111110	254	FE	11111110
63	3F	00111111	127	7F	01111111	191	BF	10111111	255	FF	11111111