intel®

EMV-51A
EMULATION VEHICLE
POCKET REFERENCE

Intel Corporation, 3065 Bowers Avenue,
Santa Clara, California 95051

Order No. 165625-001

CONTENTS

Conventions

UPPERCASE	must be entered exactly as shown.
class name	variable information.
[*option*]	optional item.
...	repeatable item.
{ *item* } { *item* }	one and only one entry must be selected.
UNDERLINE	underscored parts of command are allowable abbreviations.

Entry Editing

ESC	Deletes the command being entered and halts emulation of the user's program.
RETURN	Signifies the end of the command line.
RUB OUT	Deletes the last character that was entered.
CTRL P	Allows literal entry of control characters.
CTRL R	Redisplays current command line being entered.
CTRL X	Deletes the current line in the line-editing buffer.
CTRL Z	Deletes entire command entry.

Controlling The Display

CTRL Q	Resumes console display.
CTRL S	Stops console display.
FUNCT S	Slows down or speeds up screen output.

Terms

expr	An expression, composed of numerals, keywords, and symbolic references that is evaluated to a 16-bit number.
string	A sequence of one of more alpha-numeric characters enclosed in apostrophes.
source	Refers to the assembler mnemonic input to the command. The *source pn* is the input file.
destination	Refers to the output for the command. The *destination pn* is the output file.
command	One or a list of valid EMV-51A commands.
condition	Has the same kinds of entries as expressions, however, only the least significant bit of the result is tested.

EMV-51A Console Commands

ACCUMULATOR

Display or change the contents of the accumulator

ACC [= *expr*]

expr specify an 8-bit numeric value

Example

ACC=10H

ASM

Display current assembly location counter or assemble instruction mnemonics into program memory.

ASM [{ ORG expr / *source* }]

ORG *expr* — change the contents of the assembly location counter to the value of *expr*.

source — specify a valid 8051/8751/8031 instruction mnemonic.

Examples

```
ASM
ASM MOV R0,#7
```

B

Display or change the contents of the B register.

B [= *expr*]

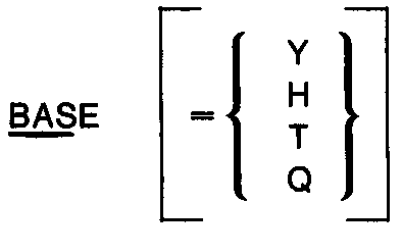
expr specify the 8-bit numeric value

Example

```
B=10H
```

BASE

Display or change the numeric data display base.



Y binary as the base.
H hexadecimal as the base.
T decimal as the base.
Q octal as the base.

Example

BAS=T

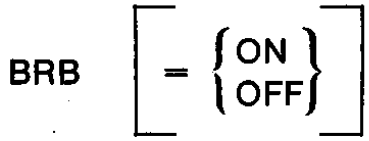
BC

Clear all breakpoints

BC

BRB

Enable, disable, or display the branch breakpoint.



Examples

BRB=ON
BRB=OFF

BREAK

Display all break commands and values.

BREAK

BRR

Specify or display a range breakpoint.

BRR [= { *expr1* TO *expr2* / OFF }]

expr1 beginning address within the range of addresses

expr2 ending address within the range of addresses.

OFF disable the range breakpoint.

Examples

```
BRR=100H TO 150H
BRR=.ABLE+4 TO .SETUP-2
```

BR*n*

Specify or display address breakpoints.

BR*n* [= { *expr* / OFF }]

n numeric value between 0 and 3 that specifies breakpoint register.

expr specify 16-bit address for execution breakpoint.

OFF disable specified address breakpoint.

Examples

```
BR0=100H
BR2=.ABLE+5
```

BV

Specify or display a value breakpoint.

BR [= { *register expr* | OFF }]

register 8051 registers R0-R7 and ACC.

expr cause a break when the register achieves the breakpoint value.

OFF disable the value breakpoint.

Example

BV=ACC 87H

CBYTE

Display or modify contents of program memory

CBYTE *expr1* [{ TO *expr2* | LENGTH *expr4* }] [= { *expr3* | *string* } [,*expr* | ,*string*] ...]

expr1 beginning address in program memory.

TO *expr2* ending address in program memory.

LENGTH *expr4* range of locations in program memory to be operated upon.

expr3 8-bit value that is stored at the specified address(es).

string alphanumeric string of characters that are stored beginning at the specified address.

Examples

CBY 0 to 3FH
CBY 0 LEN 9 = 1,2,CBY 56H

CDUMP

Display program memory as hexadecimal and ASCII.

CDUMP *expr1* TO *expr2*

expr1	beginning address of a range of addresses to be displayed.
TO *expr2*	ending address of a range of addresses to be displayed.

Examples

```
CD 80H TO 83H
CD .ABLE TO .FIN
```

COUNT

Begin a finite command loop.

```
COUNT expr <cr>
    [ WHILE condition <cr>
      UNTIL condition <cr>   ...
      command <cr> ]
END
```

expr	number of times the loop is repeated.
WHILE	loop terminates early when the least significant bit tested is false.
UNTIL	loop terminates early when the least significant bit tested is true.
condition	test condition that satisfies the WHILE/ UNTIL clauses.
command	one or a sequence of EMV-51A commands.

Examples

```
COU 5
   WHILE DBYTE 0 < .TOTAL
   command
END

COUNT 10T
   UNTIL DBYTE 0=.TOTAL
   command
END
```

DASM

Display program memory as instruction mnemonics.

DASM *expr1* TO *expr2*

expr1	beginning address of the range of addresses to be disassembled.
TO *expr2*	end address of a range of addresses to be disassembled.

Examples

```
DASM 100H TO 150H
DAS .ABLE TO .BAKER
```

DBYTE

Display or modify contents of data memory.

DBYTE *expr1* [{ TO *expr2* / LENGTH *expr4* }] [= { *expr3* / *string* } [,*expr* / ,*string*] ...]

expr1	beginning address in data memory.
TO *expr2*	ending address between 0 and 7FH.
LENGTH *expr4*	range of locations between 0 and 7FH.
=	data memory locations are changed to following value.
expr3	8-bit value stored at specified address.
string	alphanumeric string of characters that are stored at the specified address.

Examples

```
DBY 0 TO 3FH
DBY 0 LEN 9 = 56H
```

DDUMP

Display data memory in hexadecimal and ASCII.

DDUMP *expr1* TO *expr2*

expr1 beginning address of range of addresses.

TO *expr2* ending address of range of addresses.

Examples

```
DD 8H TO 4AH
DD.ABLE+4 TO .SEC+10H
```

DEFINE

Define symbolic names or macro definitions.

DEFINE { *:string* | *.string = expr* }

:string macro definition where *string* is an alphanumeric string of 2 to 32 characters, including the colon, that is the name of a new macro.

.string alphanumeric string of 2 to 32 characters, including the period, that becomes a user symbolic name or macro name.

expr 16-bit value to be assigned to the symbol name.

Examples

```
DEF.CHALK=.ABLE+FFH
DEF:NULL
```

DIR

Displays the name of macros.

DIR

DISABLE

Disable the display of symbol names or macro text.

DISABLE { SYMBOLIC | EXPANSION }

SYMBOLIC disable display of symbolic names.

EXPANSION disable display of macro definitions

DPTR

Display or change contents of data pointer register.

DPTR[=*expr*]

expr 16-bit address value to be assigned to the data pointer register.

Examples

```
DPTR
DPTR=100
H
```

DTRACE

Display table of trace commands and current setting.

DTRACE

ENABLE

Enable the display of symbol names or macro text.

ENABLE { SYMBOLIC | EXPANSION }

SYMBOLIC enable display of symbolic names.

EXPANSION enable display of macro definitions

EVALUATE

Evaluate an expression.

EVALUATE *expr*

expr 16-bit arithmetic expression or symbolic name.

Examples

```
EVA 30H+23*(.ABLE+4)
EVA.ABLE
```

EXIT

Terminate debugging session and return to ISIS-iPDS.

EXIT

FUNCTION

Display or assign function keys.

FUNCTION [= *n* : *string*]

n a numeric value between 0 and 9 inclusive.

:string a pre-defined macro name that consists of 2 to 9 alphanumeric characters including the colon.

Examples

```
FUN=5:MULT
FUN
```

GO

Begin full speed emulation of the user's program.

GO [[FROM] *expr*]

expr 16-bit address where emulation is to being.

Examples

GO
G FROM 100H

HELP

Display information about EMV-51A commands.

HELP [*item*][,...]

item specify an EMV-51A command or keyword.

Examples

HELP
HELP LOAD, LIST

IF..THEN..ELSE

Conditionally execute a series of commands.

```
IF cond1 [THEN] <cr>
   [command <cr>...]
   [ORIF cond2 <cr>
   command <cr>...] ...
   [ELSE <cr>
   command <cr>] ...
ENDIF
```

cond1 conditional test whose 16-bit result is true or false.

cond2 conditional test whose 16-bit result is true or false.

command one or a list of EMV-51A commands to be executed if the test condition is true.

Examples

```
IF DBYTE 100=1
   STEP FROM 100
ELSE
   CBYTE 0 TO 200
ENDIF

IF.G=0 THEN
      STEP FROM 100H
   ORIF .G=1 THEN
      .ABLE=.ABLE+1
      STEP FROM 200H
   ORIF .G=.ABSCAN THEN
      CBYTE 0 TO 300H
      .G=.G+1
ELSE
      GO FROM 50H
ENDIF
```

INCLUDE

Load a macro definition or command file.

INCLUDE *source pn*

source pn specify the disk number and file name of the macro or command file.

Examples

```
INC:F1:MAC.SAV
INC FILE.SAV
```

INTERRUPT

Displays a table of the interrupt indicators.

INTERRUPT

LIST

Send a copy of the debugging session to disk file.

LIST *destination pn*

destination pn specify the output file that receives a copy of the debugging session.

Examples

```
LIST :F2:TEST.V3
LIST :SO:
```

LOAD

Load the user's object code and/or symbol table.

LOAD *source pn* [NOCODE] [NOSYMBOL]

source pn	input a file that contains the object code and symbol table of the user's program.
NOCODE	specify that the user's object code is not loaded.
NOSYMBOL	specify that the user's symbol table is not loaded.

Examples

```
LOA :F1:EMVDEM.OBJ
LOA :F4:EMVDEM.OBJ NOSYMBOL
```

MACRO

Display the text definition of one or more macros.

MACRO [:*string1* [,:*string2*,...]]

:*string*	a macro name that is 2 to 32 alphanumeric characters including the colon.

Examples

```
MAC
MAC :SUM,:DIV
```

MEMORY

Display formats for display/modify commands.

MEMORY

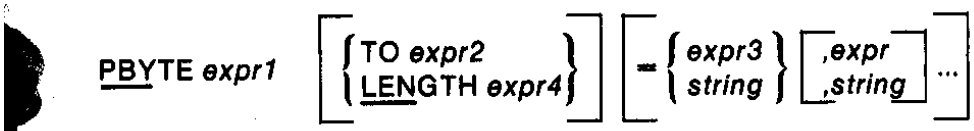
P

Display the last two instructions emulated.

P

PBYTE

Display or modify contents of external data memory.



expr1	beginning address in external data memory.
TO *expr2*	ending address in external data memory between 0 and FFFFH.
LENGTH *expr4*	range of locations in external data memory to be displayed between 0 and FFFFH.
expr3	8-bit value that is stored at specified address.
string	alphanumeric string of characters that are stored in memory at specified address.

Examples

```
PBY 0 TO 3FH
PBY 0 LEN 9 = 56H
```

PC

Display or change contents of program counter.

PC [= *expr*]

expr	16-bit memory address that replaces the current contents of the program counter.

Examples

```
PC
PC=100H
```

PSW

Display or change program status register (PSW).

PSW [=*expr*]

expr 8-bit numeric value that replaces the current contents of the PSW.

Examples

```
PSW
PSW=1
```

PUT

Save the definition of user macros in a file.

PUT *destination pn* { *;string* [,*:string*,...] | MACRO }

destination pn output file that receives a copy of the macro definition.

:string a pre-defined macro name that consists of 2 to 32 alphanumeric characters including the colon.

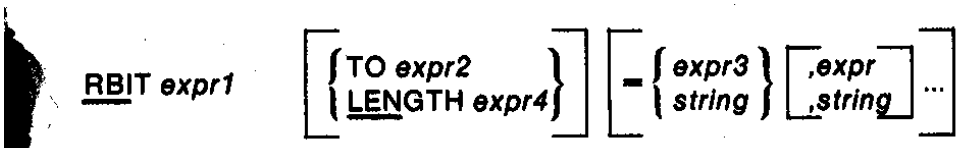
MACRO specify all macros.

Examples

```
PUT :F2:MAC.SAV :DIV,:SUM
PUT :LP: MACRO
```

RBIT

Display or modify bit-addressable memory.



expr1	beginning address in bit-addressable data memory
TO *expr2*	ending address in bit-addressable data memory between 0H and FFH.
LENGTH *expr4*	range of locations in bit-addressable data memory between 0H and FFH.
expr3	1-bit value that is stored at the specified address
string	alphanumeric string of characters whose least significant bit is stored in successive memory locations.

Examples

```
RBI 20H TO 27H
RBI D0H TO D7H = 1
```

RBS

Display or change register bank select flags.

RBS [= *expr*]

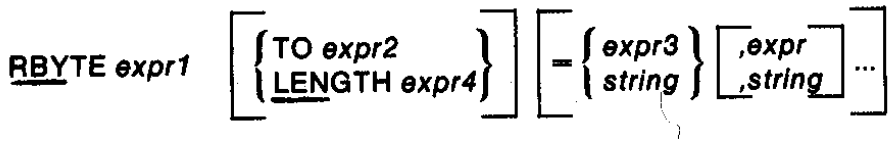
expr	numeric value, between 0 and 3, that replaces the contents of register bank select flags.

Examples

```
RBS
RBS=1
```

RBYTE

Display or modify contents of register memory.



expr1	beginning address in register memory.
TO *expr2*	ending address in register memory between 80H and FFH.
LENGTH *expr4*	range of locations in register memory that is between 0 and 7FH.
expr3	8-bit value that is stored at the specified address
string	alphanumeric string of characters stored in memory beginning at the specified address.

Examples

```
RBY 80H TO 83H
RBY 84H LEN4 = 2
```

REGISTER

Display the 8051 data registers and their contents

REGISTER

R*n*

Display or change the general purpose registers.

R*n* [=*expr*]

n specify one of the general purpose registers R0 through R7 in the register bank selected via RBS.

expr 8-bit numeric value that replaces the contents of general purpose register.

Examples

```
R0
R2=100H
```

REMOVE

Remove user-symbolic names or macro definitions.

REMOVE { *.string* [,*.string*,...] | *:string* [,*:string*,...] | SYMBOLS | MACRO }

.string name of user-defined symbol that is 2 to 32 characters long including the period.

:string name of user-defined macro that is 2 to 32 characters long including the colon.

SYMBOLS specify all user symbolic names be removed from the user symbol table.

MACRO specify all macro definitions be removed from RAM or the workfiles.

Examples

```
REM.ABLE,.BAKER
REM:DIV
```

REPEAT

Begin an infinite command loop.

```
REPEAT <cr>
   [ WHILE condition <cr>
     UNTIL condition <cr>   ...
     command <cr>          ]
END
```

WHILE	the *command* executes as long as the *condition* is true.
UNTIL	the *command* executes as long as the *condition* is false.
condition	test condition that satisfies the WHILE/UNTIL clauses.
command	one or a sequence of EMV-51A commands entered by the user.

Examples

```
REPEAT
   WHILE DBYTE 0 < .TOTAL
   ASM MOV .P0,#(DBY 0)
   DBY 0=(DBY 0+1)
END

REPEAT
   UNTIL DBYTE 0 = .TOTAL
   ASM MOV,#(DBY 0)
   DBY 0=(DBY 0+1)
END
```

RESET

Reset emulator software to its default settings.

```
RESET
```

SAVE

Save the user's program in a file.

SAVE *destination pn* [*expr1* TO *expr2* | NOCODE] [NOSYMBOL]

destination pn	output file that receives a copy of the user's program.
expr1 TO *expr2*	specific range of the user's program that is saved in the output file.
NOCODE	specify that the user's program code is not saved.
NOSYMBOL	specify that the user's symbol table is not saved.

Examples

```
SAV:F3:EMVDEM.S3 100H TO 1FFH
SAV :F4:EMVDEM.S4 NOSYMBOL
```

SP

Display or change the contents of the stack pointer.

SP [= *expr*]

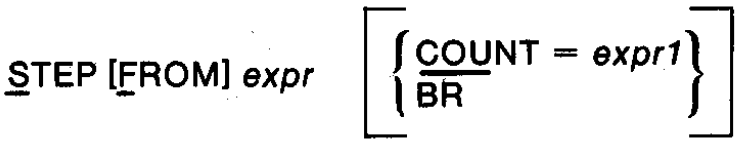
expr	8-bit memory address that replaces the current contents of the stack pointer.

Examples

```
SP
SP=F0H
```

STEP

Begin step emulation of user's program.



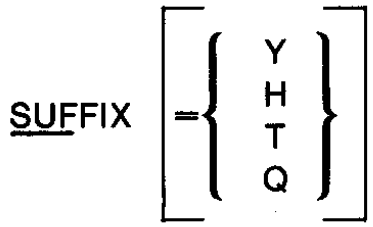
[FROM] *expr*	optional 16-bit starting address.
COUNT = *expr1*	optional number of instructions to be emulated between 0 and FFFFH.
BR	enables software breakpoint checking.

Examples

```
STEP FROM 100 BR
S F 100H COU=10
```

SUFFIX

Display or change the radix for input numeric data.



Y	binary as the default suffix.
H	hexadecimal as the default suffix.
T	decimal as the default suffix.
Q	octal as the default suffix.

Examples

```
SUF
SUF=T
```

SYMBOLS

Display user-assigned symbolic names and values.

SYMBOLS

TB*n*

Enables or disables the display of up to four one-bit memory locations.

TB*n* [= { ON / OFF }]

n specifies one of four trace bit commands (0 through 3).

Examples

TB0
TB0=4H

TD

Enables or disables the display of the disassembled instruction.

TD [= { ON / OFF }]

Examples

TD
TD=ON

TM*n*

Display or change timer/counter register 0.

TM*n* [=*expr*]

expr specifies the 16-bit numeric value that replaces the current contents of the timer/counter register.

n specifies timer/counter register 0 or timer/counter register 1. The current value of the specified timer/counter register can be displayed or changed.

Examples

```
TM0
TM0 =100H
```

TR

Enables or disables the display of general purpose registers.

TR [= {ON / OFF}]

Examples

```
TR
TR=ON
```

TR*n*

Enables or disables the trace display.

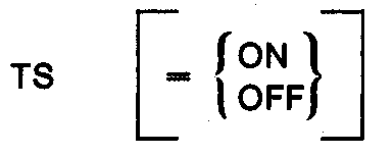
TR*n* [= *expr* {ON / OFF}]

Examples

```
TR0
TR0=100H ON
```

TS

Enables or disables the display of the program status word register.

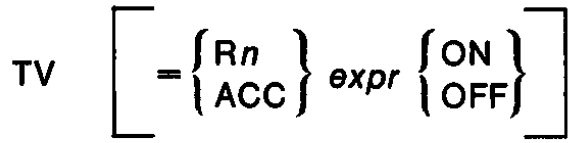


Examples

```
TS
TS=ON
```

TV

Enables or disables the trace display.



Rn specifies registers R0 through R7.

expr specifies a value that will be contained in either the specified register or the accumulator.

Examples

```
TV=R0 44H ON
TV=ACC 10H OFF
```

WRITE

Evaluate an expression and displays the results.

string a sequence of alphanumeric characters.

expr arithmetic expression or a numeric value.

Examples

```
WRITE 'BOARDS"S BAD"
```

(macro execution)

Execute macro definitions

:string [parameter1] [,parameter2,...]

:string macro name that is 2 to 32 alphanumeric characters long including the colon.

parameter specifies any parameter values required within the macro definition.

Examples

```
:SUM,:DIV
:DEMO 123,12,.ABLE
```

(symbol handling)

Display system and user symbol names or modifies user symbol names.

.string = expr

.string symbolic name that is 2 to 32 alphanumeric characters including the period.

expr 16-bit value assigned to the symbolic name.

Examples

```
.ABLE
.ABLE=123H
```

Summary of EMV-51A Errors by Number

24 Too many breaks
25 Relocatable file
26 Unresolved externals
52 Unwriteable memory
80 Syntax Error
81 Invalid Token
83 Inappropriate Number
84 Partition Bounds Error
85 Item already exists
86 Item does not exist
88 Macro parameter error
89 Missing CR-LF in file
8F Non-null string needed
90 Memory overflow
92 Command too long
94 Non-changeable item
95 Invalid object file
99 Excessive iterated data
9D Line too long
A4 Macro file full
B3 Offset is too large
B8 Assembly impossible
B9 No help available
BC System symbol error
CB Truncated to 8 bits
CD Truncated to 11 bits
E7 Illegal filename
E8 Illegal device
F0 No such file
F1 Write-protected file
F3 Checksum error
F6 Diskette file required
F9 Illegal access
FA No file name
FF Null file extension

Hexadecimal to Decimal Conversion

MOST SIGNIFICANT BYTE				LEAST SIGNIFICANT BYTE			
DIGIT 4		DIGIT 3		DIGIT 2		DIGIT 1	
HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4 096	1	256	1	16	1	1
2	8 192	2	512	2	32	2	2
3	12 288	3	768	3	48	3	3
4	16 384	4	1 024	4	64	4	4
5	20 480	5	1 280	5	80	5	5
6	24 576	6	1 536	6	96	6	6
7	28 672	7	1 792	7	112	7	7
8	32 768	8	2 048	8	128	8	8
9	36 864	9	2 304	9	144	9	9
A	40 960	A	2 560	A	160	A	10
B	45 056	B	2 816	B	176	B	11
C	49 152	C	3 072	C	192	C	12
D	53 248	D	3 328	D	208	D	13
E	57 344	E	3 548	E	224	E	14
F	61 440	F	3 840	F	240	F	15
7654		3210		7654		3210	
BYTE				BYTE			

Base Conversions

DEC	BIN	HEX	OCT	DEC	BIN	HEX	OCT
0	0000 0000	00	000	51	0011 0011	33	063
1	0000 0001	01	001	52	0011 0100	34	064
2	0000 0010	02	002	53	0011 0101	35	065
3	0000 0011	03	003	54	0011 0110	36	066
4	0000 0100	04	004	55	0011 0111	37	067
5	0000 0101	05	005	56	0011 1000	38	070
6	0000 0110	06	006	57	0011 1001	39	071
7	0000 0111	07	007	58	0011 1010	3A	072
8	0000 1000	08	010	59	0011 1011	3B	073
9	0000 1001	09	011	60	0011 1100	3C	074
10	0000 1010	0A	012	61	0011 1101	3D	075
11	0000 1011	0B	013	62	0011 1110	3E	076
12	0000 1100	0C	014	63	0011 1111	3F	077
13	0000 1101	0D	015	64	0100 0000	40	100
14	0000 1110	0E	016	65	0100 0001	41	101
15	0000 1111	0F	017	66	0100 0010	42	102
16	0001 0000	10	020	67	0100 0011	43	103
17	0001 0001	11	021	68	0100 0100	44	104
18	0001 0010	12	022	69	0100 0101	45	105
19	0001 0011	13	023	70	0100 0110	46	106
20	0001 0100	14	024	71	0100 0111	47	107
21	0001 0101	15	025	72	0100 1000	48	110
22	0001 0110	16	026	73	0100 1001	49	111
23	0001 0111	17	027	74	0100 1010	4A	112
24	0001 1000	18	030	75	0100 1011	4B	113
25	0001 1001	19	031	76	0100 1100	4C	114
26	0001 1010	1A	032	77	0100 1101	4D	115
27	0001 1011	1B	033	78	0100 1110	4E	116
28	0001 1100	1C	034	79	0100 1111	4F	117
29	0001 1101	1D	035	80	0101 0000	50	120
30	0001 1110	1E	036	81	0101 0001	51	121
31	0001 1111	1F	037	82	0101 0010	52	122
32	0010 0000	20	040	83	0101 0011	53	123
33	0010 0001	21	041	84	0101 0100	54	124
34	0010 0010	22	042	85	0101 0101	55	125
35	0010 0011	23	043	86	0101 0110	56	126
36	0010 0100	24	044	87	0101 0111	57	127
37	0010 0101	25	045	88	0101 1000	58	130
38	0010 0110	26	046	89	0101 1001	59	131
39	0010 0111	27	047	90	0101 1010	5A	132
40	0010 1000	28	050	91	0101 1011	5B	133
41	0010 1001	29	051	92	0101 1100	5C	134
42	0010 1010	2A	052	93	0101 1101	5D	135
43	0010 1011	2B	053	94	0101 1110	5E	136
44	0010 1100	2C	054	95	0101 1111	5F	137
45	0010 1101	2D	055	96	0110 0000	60	140
46	0010 1110	2E	056	97	0110 0001	61	141
47	0010 1111	2F	057	98	0110 0010	62	142
48	0011 0000	30	060	99	0110 0011	63	143
49	0011 0001	31	061	100	0110 0100	64	144
50	0011 0010	32	062	101	0110 0101	65	145

Base Conversions (Continued)

DEC	BIN	HEX	OCT	DEC	BIN	HEX	OCT
102	0110 0110	66	146	153	1001 1001	99	231
103	0110 0111	67	147	154	1001 1010	9A	232
104	0110 1000	68	150	155	1001 1011	9B	233
105	0110 1001	69	151	156	1001 1100	9C	234
106	0110 1010	6A	152	157	1001 1101	9D	235
107	0110 1011	6B	153	158	1001 1110	9E	236
108	0110 1100	6C	154	159	1001 1111	9F	237
109	0110 1101	6D	155	160	1010 0000	A0	240
110	0110 1110	6E	156	161	1010 0001	A1	241
111	0110 1111	6F	157	162	1010 0010	A2	242
112	0111 0000	70	160	163	1010 0011	A3	243
113	0111 0001	71	161	164	1010 0100	A4	244
114	0111 0010	72	162	165	1010 0101	A5	245
115	0111 0011	73	163	166	1010 0110	A6	246
116	0111 0100	74	164	167	1010 0111	A7	247
117	0111 0101	75	165	168	1010 1000	A8	250
118	0111 0110	76	166	169	1010 1001	A9	251
119	0111 0111	77	167	170	1010 1010	AA	252
120	0111 1000	78	170	171	1010 1011	AB	253
121	0111 1001	79	171	172	1010 1100	AC	254
122	0111 1010	7A	172	173	1010 1101	AD	255
123	0111 1011	7B	173	174	1010 1110	AE	256
124	0111 1100	7C	174	175	1010 1110	AF	257
125	0111 1101	7D	175	176	1011 0000	B0	260
126	0111 1110	7E	176	177	1011 0001	B1	261
127	0111 1111	7F	177	178	1011 0010	B2	262
128	1000 0000	80	200	179	1011 0011	B3	263
129	1000 0001	81	201	180	1011 0100	B4	264
130	1000 0010	82	202	181	1011 0101	B5	265
131	1000 0011	83	203	182	1011 0110	B6	266
132	1000 0100	84	204	183	1011 0111	B7	267
133	1000 0101	85	205	184	1011 1000	B8	270
134	1000 0110	86	206	185	1011 1001	B9	271
135	1000 0111	87	207	186	1011 1010	BA	272
136	1000 1000	88	210	187	1011 1011	BB	273
137	1000 1001	89	211	188	1011 1100	BC	274
138	1000 1010	8A	212	189	1011 1101	BD	275
139	1000 1011	8B	213	190	1011 1110	BE	276
140	1000 1100	8C	214	191	1011 1111	BF	277
141	1000 1101	8D	215	192	1100 0000	C0	300
142	1000 1110	8E	216	193	1100 0001	C1	301
143	1000 1111	8F	217	194	1100 0010	C2	302
144	1001 0000	90	220	195	1100 0011	C3	303
145	1001 0001	91	221	196	1100 0100	C4	304
146	1001 0010	92	222	197	1100 0101	C5	305
147	1001 0011	93	223	198	1100 0110	C6	306
148	1001 0100	94	224	199	1100 0111	C7	307
149	1001 0101	95	225	200	1100 1000	C8	310
150	1001 0110	96	226	201	1100 1001	C9	311
151	1001 0111	97	227	202	1100 1010	CA	312
152	1001 1000	98	230	203	1100 1011	CB	313

Base Conversions (Continued)

DEC	BIN	HEX	OCT	DEC	BIN	HEX	OCT
204	1100 1100	CC	314	230	1110 0110	E6	346
205	1100 1101	CD	315	231	1110 0111	E7	347
206	1100 1110	CE	316	232	1110 1000	E8	350
207	1100 1111	CF	317	233	1110 1001	E9	351
208	1101 0000	D0	320	234	1110 1010	EA	352
209	1101 0001	D1	321	235	1110 1011	EB	353
210	1101 0010	D2	322	236	1110 1100	EC	354
211	1101 0011	D3	323	237	1110 1101	ED	355
212	1101 0100	D4	324	238	1110 1110	EE	356
213	1101 0101	D5	325	239	1110 1111	EF	357
214	1101 0110	D6	326	240	1111 0000	F0	360
215	1101 0111	D7	327	241	1111 0001	F1	361
216	1101 1000	D8	330	242	1111 0010	F2	362
217	1101 1001	D9	331	243	1111 0011	F3	363
218	1101 1010	DA	332	244	1111 0100	F4	364
219	1101 1011	DB	333	245	1111 0101	F5	365
220	1101 1100	DC	334	246	1111 0110	F6	366
221	1101 1101	DD	335	247	1111 0111	F7	367
222	1101 1110	DE	336	248	1111 1000	F8	370
223	1101 1111	DF	337	249	1111 1001	F9	371
224	1110 0000	E0	340	250	1111 1010	FA	372
225	1110 0001	E1	341	251	1111 1011	FB	373
226	1110 0010	E2	342	252	1111 1100	FC	374
227	1110 0011	E3	343	253	1111 1101	FD	375
228	1110 0100	E4	344	254	1111 1110	FE	376
229	1110 0101	E5	345	255	1111 1111	FF	377

Powers of Two and Sixteen

Powers of Two

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

Conversion Between Powers of 2 and 16

$2^m = 16^n$
$2^0 = 16^0$
$2^4 = 16^1$
$2^8 = 16^2$
$2^{12} = 16^3$
$2^{16} = 16^4$
$2^{20} = 16^5$
$2^{24} = 16^6$
$2^{28} = 16^7$
$2^{32} = 16^8$
$2^{36} = 16^9$
$2^{40} = 16^{10}$
$2^{44} = 16^{11}$
$2^{48} = 16^{12}$
$2^{52} = 16^{13}$
$2^{56} = 16^{14}$
$2^{60} = 16^{15}$
$2^{64} = 16^{16}$

Powers of Sixteen

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

ASCII Code List

Dec.	Oct.	Hex.	Char.	Dec.	Oct.	Hex.	Char.
0	000	00	NUL	47	057	2F	/
1	001	01	SOH	48	060	30	0
2	002	02	STX"	49	061	31	1
3	003	03	ETX	50	062	32	2
4	004	04	EOT	51	063	33	3
5	005	05	ENQ	52	064	34	4
6	006	06	ACK	53	065	35	5
7	007	07	BEL	54	066	36	6
8	010	08	BS	55	067	37	7
9	011	09	HT	56	070	38	8
10	012	0A	LF	57	071	39	9
11	013	0B	VT	58	072	3A	:
12	014	0C	FF	59	073	3B	;
13	015	0D	CR	60	074	3C	<
14	016	0E	SO	61	075	3D	=
15	017	0F	SI	62	076	3E	>
16	020	10	DLE	63	077	3F	?
17	021	11	DC1	64	100	40	@
18	022	12	DC2	65	101	41	A
19	023	13	DC3	66	102	42	B
20	024	14	DC4	67	103	43	C
21	025	15	NAK	68	104	44	D
22	026	16	SYN	69	105	45	E
23	027	17	ETB	70	106	46	F
24	030	18	CAN	71	107	47	G
25	031	19	EM	72	100	48	H
26	032	1A	SUB	73	101	49	I
27	033	1B	ESC	74	102	4A	J
28	034	1C	FS	75	103	4B	K
29	035	1D	GS	76	104	4C	L
30	036	1E	RS	77	105	4D	M
31	037	1F	US	78	106	4E	N
32	040	20	SP	79	107	4F	O
33	041	21	!	80	100	50	P
34	042	22	"	81	101	51	Q
35	043	23	#	82	102	52	R
36	044	24	$	83	103	53	S
37	045	25	%	84	104	54	T
38	046	26	&	85	105	55	U
39	047	27	'	86	106	56	V
40	050	28	(	87	107	57	W
41	050	29	)	88	100	58	X
42	052	2A	*	89	101	59	Y
43	053	2B	+	90	102	5A	Z
44	054	2C	,	91	103	5B	[
45	055	2D	-	92	104	5C	\
46	056	2E	.	93	105	5D	]

ASCII Code List (Continued)

Dec.	Oct.	Hex.	Char.	Dec.	Oct.	Hex.	Char.
94	106	5E	^	111	157	6F	o
95	107	5F	_	112	160	70	p
96	140	60	`	113	161	71	q
97	141	61	a	114	162	72	r
98	142	62	b	115	163	73	s
99	143	63	c	116	164	74	t
100	144	64	d	117	165	75	u
101	145	65	e	118	166	76	v
102	146	66	f	119	167	77	w
103	147	67	g	120	170	78	x
104	150	68	h	121	171	79	y
105	151	69	i	122	172	7A	z
106	152	6A	j	123	173	7B	{
107	153	6B	k	124	174	7C	|
108	154	6C	l	125	175	7D	}
109	155	6D	m	126	176	7E	~
110	156	6E	n	127	177	7F	DEL

ASCII Code in Binary

MSB / LSB	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL	DLE	SP	0	@	P	`	p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	$	4	D	T	d	t
5 0101	ENQ	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	^	n	~
F 1111	SI	VS	/	?	O	_	o	DEL

ASCII Code Definition

Abbreviation	Meaning	Decimal Code
NUL	NULL Character	0
SOH	Start of Heading	1
STX	Start of Text	2
ETX	End of Text	3
EOT	End of Transmission	4
ENQ	Enquiry	5
ACK	Acknowledge	6
BEL	Bell	7
BS	Backspace	8
HT	Horizontal Tabulation	9
LF	Line Feed	10
VT	Vertical Tabulation	11
FF	Form Feed	12
CR	Carriage Return	13
SO	Shift Out	14
SI	Shift In	15
DLE	Data Link Escape	16
DC1	Device Control 1	17
DC2	Device Control 2	18
DC3	Device Control 3	19
DC4	Device Control 4	20
NAK	Negative Acknowledge	21
SYN	Synchronous Idle	22
ETB	End of Transmission Block	23
CAN	Cancel	24
EM	End of Medium	25
SUB	Substitute	26
ESC	Escape	27
FS	File Separator	28
GS	Group Separator	29
RS	Record Separator	30
US	Unit Separator	31
SP	Space	32
DEL	Delete	127

3065 Bowers Avenue, Santa Clara, California 95051
(408) 987-8080
Printed in U.S.A.

Development Systems

0115/3K/1284/AP/AD