ERROR MESSAGES

0 — ! UNDEFINED ERROR CODE !
1 — I/O ERROR, OS ERROR CODE RETURNED
2 — INSUFFICIENT MEMORY TO CONTINUE
3 — ! SEGMENT VIOLATION !
4 — I/O ERROR INVALID UNIT ID
5 — I/O ERROR READ/WRITE VIOLATION
6 — I/O ERROR INSUFFICIENT MEMORY FOR OPEN
7 — ! DELETE UNIT CONTROL BLOCKS ERROR !
8 — TOO MANY IDT DEF/REF SYMBOLS IN LOAD
9 — EXCEEDED 15 LOAD OPERATIONS SINCE LAST CLR
10 — CANNOT ALLOCATE MEMORY FOR USER SYMBOL TABLE
11 — ! ERROR IN I/O UNIT CHAIN POINTERS !
12 — OVERLAY ERROR
101 — VARIABLE CANNOT BE READ
102 — VARIABLE CANNOT BE WRITTEN
103 — SYMBOL IS UNDEFINED
104 — ! INVALID CODEGEN BRANCH TABLE INDEX !
105 — INSUFFICIENT MEMORY TO COMPILE STATEMENT
106 — SYMBOL IS DEFINED, CANNOT BE REDEFINED
107 — INSUFFICIENT MEMORY TO COMPILE PROC/FUNC
108 — INPUT RECORD CANNOT BE CLASSIFIED
109 — INPUT STRING EXCEEDS MAXIMUM ALLOWED LENGTH
110 — ! INVALID SCANNER BRANCH TABLE INDEX !
111 — UNRECOGNIZABLE INPUT ITEM
112 — ! UNDEFINED OPERATOR !
114 — SYMBOL NOT AN IDT/DEF/REF LOAD SYMBOL
115 — USER SYMBOL TABLE FULL
116 — CONSTANT EXCEEDS 16 BITS
117 — SYNTAX ERROR
118 — ! INVALID KEYWORD STRING LENGTH !
119 — SYNTAX ERROR IN ONE-LINE-ASSEMBLY STATEMENT
120 — INCORRECT NUMBER OF ARRAY SUBSCRIPTS
121 — ESCAPE SPECIFIED OUTSIDE A LOOP CONSTRUCT
122 — ARRAY REDEFINED WITH INCORRECT SUBSCRIPTS

NOTE A hexadecimal number is also printed with some error messages Refer to the AMPL System Operation Guide for complete explanation

ERROR MESSAGES

201 — SYMBOL NOT FOUND TO DELETE
202 — SYMBOL CANNOT BE DELETED
203 — INVALID DISPLAY FORMAT CHARACTER FOLLOWING
204 — NO LIST DEVICE ASSIGNED
205 — EMULATOR I/O ERROR CODE RETURNED
209 — INVALID INDEX INTO EMULATOR TRACE BUFFER
210 — !CANNOT ALLOCATE FORM CURRENT VALUE SEGMENT!
211 — INSUFFICIENT MEMORY TO SAVE FORM PARAMETERS
214 — INVALID RESTORE FILE
215 — INSUFFICIENT MEMORY TO COMPLETE THE RESTORE
216 — BAD TRACE OR COMPARISON MODE SELECTED
219 — TRACE MODULE I/O ERROR CODE RETURNED
220 — CANNOT EDIT ON THIS DEVICE TYPE
221 — TRACE INTERFACE CHANGE ILLEGAL WHILE TRACING
222 — INVALID INDEX INTO TRACE MODULE BUFFER
223 — INSUFFICIENT ARGUMENTS IN PROC/FUNC CALL
224 — STACK OVERFLOW; DELETE PROC/FUNC/ARRAY
225 — DELETED PROC/FUNC/ARRAY REFERENCED
226 — INSUFFICIENT ARGUMENTS IN FORM FOR PROC/FUNC
227 — ! INVALID FORM SEGMENT ID !
228 — ! INVALID FORM CURRENT VALUE SEGMENT ID !
229 — INVALID CHARACTER IN LOAD FILE
230 — CHECKSUM ERROR IN LOAD FILE
231 — ARITHMETIC OVERFLOW
233 — PROC/FUNC CALL ARGUMENT OUT OF RANGE
234 — INVALID "ARG" OR "LOC" INDEX FOR WRITING
235 — INVALID "ARG" OR "LOC" INDEX FOR READING
237 — ARRAY ALREADY DEFINED
238 — INVALID ARRAY DIMENSION
240 — REFERENCE TO UNDECLARED ARRAY
241 — INVALID ARRAY SUBSCRIPT
242 — ! ERROR ARRAY SEGMENT LENGTH !
243 — DELETED IDT/DEF/REF LOAD SYMBOL REFERENCED
244 — ALL IDT/DEF/REF LOAD SYMBOLS DELETED
245 — INVALID DEVICE TYPE TO "EINT" OR "TINT"

NOTE Error messages withing exclamation marks (!) are AMPL internal system errors Contact Texas Instruments if problem persists

TEXAS INSTRUMENTS

AMPL Reference Data

Microprocessor Series™

EXPLANATION OF THE NOTATION USED IN THIS CARD

	Notation	Explanation
Optional Items	[item]	Bracketed item may be omitted
	{item 1 / item 2}	Exactly one item must be selected from the items in braces
Substitution	expr 'file'	Any expression may be used File or device name required
Repetition	item .	A list of items may be used
Required	<item>	Replace with item

CHARACTER SET

Type	Characters	Use
Special	RETURN SPACE ! " $ / ( ) * + , - . / , < = > ? @	Any printable character may be used in a quoted string RETURN terminates line and statement ":" may separate statements SPACE separates adjacent numbers and identifiers.
Numerals	0 — 9	
Letters	A — Z,a — z	

NOTE All AMPL reserved words use only upper case (UPPER CASE LOCK)

SYMBOL NAMES

Type	Example	Definition
System	RO ETRC	Up to four alphanumeric characters, all system symbols are predefined.
User-defined	USRVAR X3 BRKADR GO	Up to six alphanumeric characters, assignment defines a variable ARRAY statement defines an array. PROC/FUNC statement defines a procedure/function
Program label	IDT DEF	Up to six alphanumeric characters Period after IDT and before DEF labels, defined by LOAD command

CONSTANTS

Type	Example	Range
Decimal	10833	1 32767
Hexadecimal	02A51, >2A51	>0 >FFFF
Octal	125121	!0 !177777
Binary	· 101010010010001	0 1111111111111111
ASCII	Q	
Instruction	= XOR B1,R9 =	
Keyword	IAO	See keyword constant table

EXPRESSIONS

Type	Example	Definition
Subexpression	(expr)	
Identity	+ expr	Value of <expr>
Negation	- expr	Two's complement of <expr>
Target memory	@addr	<addr> used as word address into emulator or target memory
Proc/Func Argument	ARG expr	Argument in position <expr> of call list, ARG 0 is number of arguments in list
Proc/Func local variable	LOC expr	Word <expr> of local variable array, LOC 0 is length of local variable array
Multiplication	expr1*expr2	Signed product (warning on overflow).
Division	expr1/expr2	Signed quotient (warning on divide by zero)
Remainder	expr1 MOD EXPR2	Signed remainder of division (warning on divide by zero)
Addition	expr1 + expr2	Signed sum
Subtraction	expr1 - expr2	Signed difference
NOTE Result of relational operator is either FALSE (0) or TRUE (-1)		
Equality	expr1 EQ expr2 expr1 NE expr2	16-bit comparison
Arithmetic inequality	expr1 LT expr2 expr1 LE expr2 expr1 GT expr2 expr1 GE expr2	Signed, 16-bit comparison.
Logical inequality	expr1 LO expr2 expr1 LOE expr2 expr1 HI expr2 expr1 HIE expr2	Unsigned, 16-bit comparison
Complement	NOT expr	16-bit one's complement
Conjunction	expr1 AND expr2 expr 1 NAND expr2	16-bit boolean AND 16-bit boolean not AND
Disjunction	expr1 OR expr2 expr1 XOR expr2	16-bit boolean OR 16-bit boolean exclusive OR

NOTE: Operators are given in order of precedence, highest to lowest. Solid lines separate precedence groups, within each group, precedence is equal and evaluation is left to right. Evaluation results in a 16-bit integer value.

UNSIGNED ARITHMETIC

Syntax	Definition
MPY (expr1, expr2)	Low-order 16 bits of unsigned product <expr1> * <expr2>, high order 16 in MDR
DIV (divisor, dividend)	Unsigned quotient of 32-bit number (MDR, <dividend>) over <divisor>; remainder in MDR
MDR	High-order 16-bits of MPY product and of DIV dividend, remainder of DIV, unsigned carry of + and-

ARRAY DEFINITION

ARRAY name(expr1[,expr2]), .	User <name> (previously undefined or name of deleted array) is defined as one- or two-dimension array

DISPLAY STATEMENTS

expr[.f f]	Value of expression
'LITERAL STRING'	Literal string
add1 [TO addr2] [f f] ? [f . f]	Target memory

Formal specification/[f f]

ASCII	A	set default	G	octal	O[i]
binary	B[i]	hexadecimal	H[i]	symbolic	S
decimal	D[i]	instruction	I	unsigned	U[i]
name =	E	newline	N[i]	space	X[i]

Note
1 <= i <= 9 — field width 'i' digits, then two blanks
i = 0 — default field width, no trailing blanks
1 <= j <= 9 — repeat 'j' times
j = 0 — repeat 10 times

Response to display/modify mode(?)

forward step	RETURN, +	replace contents	<expr>
back step	-	open new address	@<addr>
exit	.	change display	f f

DISASSEMBLER

Instruction	DST	Destination address
operands	SRC	Source address

NOTE Additional instructions of the TMS9940 (DCA, DCS, LIIM, SM) wil ble correctly (# DCA *RC1 #) but will disassemble as XOP instructions See 0 specifications for details

ASSIGNMENT STATEMENTS

Type	Example	Definition
Variable	sym = expr	User-defined or writable system symbol or REF program label
Target memory	@addr = expr	Put value of <expr> at target <addr>
Proc/Func argument	ARG n = expr	Local copy of argument in position <n> of call list
Command local	LOC n = expr	Word <n> of local storage array.
Array	A[(n1[,n2])] = e	User defined array name, zero, one, or two index expressions

NOTE Precedence of @, ARG, and LOC may require parenthesis around following expression

COMPOUND STATEMENTS

Syntax	Definition
BEGIN statements END	Statements are executed sequentially. Use in place of any single statement syntax

CONTROL STATEMENTS

IF expr THEN s1 [ELSE s2]	<s1> is executed if <expr> is TRUE (nonzero) Otherwise, <s2> is executed, if included
CASE expr OF expr 1 s1 ... expr n sn [ELSE s] END	Statement <s> at first label expression <expr> equal to expr > is executed if none statement s is executed if included
WHILE expr DO statement	While <expr> is TRUE (nonzero), <statement> is executed
REPEAT statement UNTIL expr	<statement> is executed if <expr> FALSE (zero), <statement> is executed until <expr> is TRUE
FOR var = expr1 TO expr 2[BY expr3] DO statement	Value of <expr1> is assigned to <var> <statement> is executed until <var> is equal to <expr2>, <expr3> is added to <var>, and <statement> repeated. Default value of <exp3> is 1
ESCAPE	Exit from innermost enclosing WHILE, REPEAT, or FOR statement

PROCEDURE/FUNCTION/FORM DEFINITION

PROC name [(args[,locs])] statements END FUNC name [(args[,locs])] statements END	User-defined <name> (previously undefined or deleted procedure/function) is bound to <statements>. <args> is the required number of arguments <locs> is the size of local storage array
RETURN [expr]	Pass control back to calling statement. In a procedure, <expr> is ignored. In a function, value of <expr> replaces the function call in the calling expression.
FORM name 'prompt' [= ({ constant }]], { 'string' }	END <name> must be a previously defined procedure or function, semicolon required between prompts

PROCEDURE/FUNCTION CALLS

proc name [(expr,)]	User-defined or system procedure/function with list of argument expressions
func name [(expr,)]	
	Command definition determines number of arguments required. Some system commands require quoted strings as arguments

NOTE Procedure/functions with defined FORM when called with no arguments will prompt for arguments using the FORM example FORM.

COMMENTARY ENTRY

PROMPT 1 = default value
PRDMPT 2 =
PROMPT 3" =

comment, not a prompt required argument, with default value required argument, must enter value default given if value not entered

FORM control function keys

Next prompt	TAB, ↓ →FIELD SKIP, RETURN
Previous prompt	↑ ←FIELD
First prompt	HOME
Erase value	ERASE FIELD ERASE INPUT
Redisplay default	INSERT LINE
Duplicate previous value.	F4
Complete form	?
Abort form:	

INPUT/OUTPUT COMMANDS

Syntax	Definition
offset,width)	Host computer CRU base address Read host computer CRU field
offset,width,value)	Write <value> into host CRU field
CDPY ('file' | edit id)	AMPL input from 'file' AMPL input from edit buffer
LIST ('file' | OFF | ON | EOF)	Initialize listing device or file. Disable listing output Enable listing output. Close listing device or file with EOF
NL	Print newline.
unit = OPEN [('file' | edit id) [[0 | IN | OUT | O] [[0 | REWIND | EXTEND] [[SEQ | REL]]]]]	

no arguments — list all open units and edit buffers
initialize 'file'/<edit id> I/O unit
0 — device IO, file IN only
IN — for input only
OUT — for output only
IO — for input/output
REWIND — position to beginning of file
EXTEND — position to end of file
SEQ — auto-create sequential file
REL — auto-create rel-rec file

event-READ [(unit [[0 \| DIRECT] [[0 \| GRAPH] [[VDT [f row [f col [s col]]] \| SEQ [f row [f col]] \| REL [rec #]]]]])]

no arguments — read console
Read record from (unit)
0 — issue read ASCII
DIRECT — issue read direct
GRAPH — read graphics on 922 VDT
VDT — read in cursor positioning mode
f row — field start row
f col — field start column
s col — cursor start column
SEQ — read sequentially
REL — read sepecified record
rec # — record number to read
<event> / 256 = cursor column after read if VDT
<event> AND 255 = event key value if VDT,
else >00 for end of record,
-13 for end of file

value = EVAL [(unit)]	Evaluate expression in <unit>'s buffer, if no <unit>, READ/EVAL the console
DPLY [(unit)]	AMPL display unit for output to <unit>, if no <unit>, to console
okay = MOVE (from unit, to unit)	Move contents of <from unit>'s buffer to <to unit>'s buffer <okay> = 0 if moved = >FFFF if too big and not moved
REW[(unit)]	Rewind (unit) — repositions, file clears console no argument — clears console

Cursor = WRIT (unit [[0 \| DIRECT] [[0 \| GRAPH] [[VDT [f row [f col]] \| SEQ [f row [f col]] \| REL [rec #]]]])

no arguments — write console
Write record to (unit)
0 — issue write ASCII
DIRECT — issue write direct
GRAPH — write graphics on 911 VDT
VDT — write in cursor positioning mode
f row — field start row
f col — field start column
SEQ — write sequentially
REL — read specified record
rec # — record number to read
<cursor> / 256 = cursor column after write if VDT

CLSE (unit [, { EOF \| UNLOAD }] Release I/O <unit>,
EOF — write end-of-file mark
UNLOAD — unload unit

SYSTEM SYMBOLS

V — variable F — function P — procedure

CLR	P — clear	MDEL	P — symbols
CLSE	P — I/O close	MDR	V — arithmetic
COPY	P — copy	MIN	V — minutes
	V — CRU base	MOVE	F — I/O buffer
	F — CRU read	MPY	F — multiply
CROW	P — CRU write	MSYM	P — symbols
DAY	V — day	NL	P — newline
DBUF	P — delete buffer	OPEN	F — I/O open
DELE	P — delete symbol	PC	V — registers
DIV	F — divide	R0-R15	V — registers
DPLY	P — display	READ	F — I/O read
DR	P — registers	REW	P — I/O rewind
DST	V — destination	RSTR	P — restore
	P — dump	SAVE	P — save
	P — emulator	SEC	V — seconds
EMER	V — emulator	SRC	V — source
EDIT	F — edit	ST	V — register
EMT	P — emulator	TBRK	P — trace module
EMT	P — emulator	TEVT	P — trace module
EMEM	V — emulator	THLT	F — trace module
ERUN	P — emulator	TINT	P — trace module
EST	F — emulator	TNCE	V — trace module
ETB	F — emulator	TNE	V — trace module
ETBH	F — emulator	TRUN	P — trace module
ETBO	V — emulator	TST	P — trace module
ETRC	P — emulator	TTB	F — trace module
ETYP	V — emulator	TTBH	P — trace module
EVAL	F — evaluate	TTBN	V — trace module
EXIT	P — exit AMPL	TTBO	V — trace module
HCRB	V — host CRU	TTRC	P — trace module
HCRR	F — CRU read	USYM	P — user symbols
HCRW	P — CRU write	VRFY	P — verify
HR	V — hour	WAIT	F — delay AMPL
IORT	V — I/O	WP	V — register
KEEP	P — keep edit	WRIT	P — I/O write
LIST	P — list	YR	V — year
LOAD	P — load object		

EDIT

Syntax	Definition
edit id = EDIT[({ 'file' | edit id } [,record])]	Create edit buffer with 'file'. Edit existing buffer No argument creates an empty buffer
KEEP (edit id, 'file')	Save edit buffer onto 'file' and delete edit buffer
DBUF (edit id)	Delete edit buffer

EDIT CONTROL FUNCTION KEYS

Function	911 KEY	913 KEY	C[illegible]R
edit/compose mode	F7	F7	V
quit edit mode	CMD	HELP	X
roll up	F1	F1	A
roll down	F2	F2	B
set tab	F3	F3	C
clear tab	F4	F4	D
tab	TAB (shift SKIP)	TAB	I
back tab	SHIFT D	BACK TAB	T
newline	RETURN	NEWLINE	RETURN
insert line	unlabeled gray	INSERT LINE	O
delete line	ERASE INPUT	DELETE LINE	N
erase line	ERASE FIELD	CLEAR	W
truncate line	SKIP	SET	K
insert character	INS CHAR	INSERT CHAR	
delete character	DEL CHAR	DELETE CHAR	
cursor up	.		U
cursor down			J
cursor right			R
cursor left	.	.	H
top of screen	HOME	HOME	

GENERAL COMMANDS

Syntax	Definition
USYM	List all user symbols, procedures, functions, and arrays.
DELE ('name')	Delete user procedure, function, or array
SAVE ('file')	Save all user defined symbols, functions, and arrays on 'file'
RSTR ('file')	Restore user defined symbols, procedures, functions, and arrays from 'file'
CLR	Delete all user symbols, procedures, functions and arrays
MSYM	List object program labels
MDEL	Delete all object program labels
EXIT	Exit from AMPL back to operating system.

TIMING

YR	Year (1976 to 1999)
DAY	Julian day (1 to 366)
HR	Hour (0 to 23)
MIN	Minute (0 to 59)
SEC	Second (0 to 59)
WAIT (expr)	Suspend AMPL for <expr>*50 milliseconds (<expr> = 20 is one second).

TARGET MEMORY COMMANDS

EMEM	Emulator memory mapping. 9900/9980 map 8K bytes (0-> 1FFF) 9940 define RAM and ROM sizes
LOAD ('file'[,bias[,IDT] [+ DEF] [+ REF]]])	Load object program by bias and enter program labels into table
VRFY ('file' [,bias])	Verify object program, listing differences between object and target memory
DUMP ('file',low,high[,start])	Dump program from target <low> to <high> in nonrelocatable format

EMULATOR CONTROL COMMANDS

Syntax	Definition
EINT ('EM0n' [{1 | 0}[,'TM0n']])	Initialize Emulator device, clock 0 = prototype/ 1 = emulator
ECLK	Processor clock
ETYP	Processor type -1 = TMS9940 0 = SBP9900 1 = TMS9900, 2 = TMS9980
ETRC ({MA | IAQX | IAQ} [,count[,low,high]])	Trace qualifier completion break count (OFF-255), address range
EBRK ({MA | IAQ | MR | MW} [+ ILLA] [,address] .)	Address breakpoint(s) (ILLA only valid for TMS9940)
ERUN	Run emulation at PC, WP, ST
EST	Emulation status (3 LSBits) HOLD, IDLE, Running
EHLT	Halt emulation, return status
ETBH (index[,{MR | MW | IAQ}])	Indexed bus signal from buffer (TRUE if expression matches)
ETB (index)	Indexed address from trace buffer
ETBO, ETBN	Emulator Trace buffer limits. Oldest, Newest sample indices

TRACE MODULE CONTROL

Syntax	Definition
TINT ('TM0n')	Initialize trace module.
TTRC ([INT]{OFF | [±Q0] [±Q1][±Q2][±Q3] | [±IAQ][±DBIN]}[,count[,{ON | OFF}]])	Qualify data samples, trace completion counter (OFF-255), latch option on D0-D3
TEVT ({OFF | [±D0] [±D1] [±D2] [±D3] | [±IAQ] [±DBIN] | EXT}[,value[,mask]])	Qualify D0-D3 event (or EXTernal), <value> and <mask> for D4-D19.
TBRK (count [,<delay>[,INV] [+ EDGE]]])	Set event counter (OFF-FFFF), set delay counter (OFF-244), count INVerted/EDGE events
TRUN	Start Trace module tracing
TST	Trace module status (3 LSB's), event occurred, trace full, tracing
THLT	Halt trace module, return status
TNE	Number of events since last TRUN
TNCE	Number of event count overflows.
TTBH (index[,{ [±D0] [±D1][±D2] [±D3] | [±IAQ][±DBIN] }])	D0-D3 of indexed samples, (TRUE if expression matches)
TTB (<index>)	D4-D19 indexed samples (data bus)
TTBO, TTBN	Trace module trace buffer limits: Oldest, Newest sample indices.

TRACE MODULE INTERCONNECT TO EMULATOR

Q0	Memory address bit 15 (TMS9940 only)
D0	Byte memory cycle (TMS9940 only)
Q1,D1, IAQ	Instruction Acquisition
Q2,D2,DBIN	DataBusIN = MR(read), MW = -DBIN(write)
Q3	Emulator trace qualifier and range (ETRC)
D3 External Event	Emulator address breakpoint (EBRK)
D4-D19	Emulator data bus (bits 0-15)
External Clock	Emulator memory cycle clock
Control Cable	Synchronizes emulation and tracing Trace module will halt emulator for EINT ('EM0n', clock 'TM0n')

TARGET REGISTERS

PC,WP,ST	Processor registers
R0-R15	Workspace registers
DR	Display all registers

CRU READ/WRITE

CRUB	CRU interface base address
CRUR (offset,width)	Read target CRU field
CRUW (offset,width,value),	Write <value> into target CRU field

KEYWORDS

ARG	.	THEN	GE
ARRAY	.	TO	GT
BEGIN	#	UNTIL	HI
BY	LOC	WHILE	HIE
CASE	MOD	AND	LE
DO	NULL	NAND	LO
ELSE	OF	OR	LOE
END	PROC	XOR	LT
ESCAPE	REPEAT	NOT	NE
FOR	RETURN	EQ	

KEYWORD CONSTANTS

D0	EXT	IO	Q2
D1	EXTEND	MA	Q3
D2	GRAPH	MR	REF
D3	IAQ	MW	REL
DBIN	IAQX	N	REWIND
DEF	IDT	OFF	SEQ
DIRECT	ILLA	ON	UNLOAD
	IN	OUT	VDT
	INT	Q0	Y
ETBN	INV	Q1	