ALLEN-BRADLEY

1771 Control Coprocessor Error/Status Codes

Quick Reference

Using this Manual

This manual describes error codes and messages issued by the various programs, functions, and utilities associated with the 1771 control coprocessor. The following table lists the sections of this manual, shows the format of the codes or messages covered in each section, and lists the programs/functions that use these codes or messages.

This section:	On page:	Contains codes/messages returned/displayed as:	That are issued by these programs/functions:
Application Program Interface (API) Error Codes	1-1	number	API functions: BPI_, CC_, DTL_, MSG_, TAG_ (including calls from BASIC using AB_BAS)
OS-9 Error Codes	2-1	000:*xxx*	OS-9™ system calls, OS-9 utilities
C-Compiler Error Messages	3-1	text	`cc` or `xcc` compiler and preprocessor
Assembler/Linker Error Messages	4-1	text	`r68` assembler or `l68` linker
BASIC Error Codes	5-1	number	BASIC programs (at entry or run time)
Internet Error Codes	6-1	007:*xxx*	Internet functions/utilities

This manual lists only error codes and messages. For more information, see the following manuals:

Title	Publication Number
OS-9 Internet Software Reference Manual	1771-6.4.11
1771 Control Coprocessor User Manual	1771-6.5.95
OS-9 Operating System User Manual	1771-6.5.102
OS-9 BASIC User Manual	1771-6.4.103
OS-9 C Language User Manual	1771-6.5.104
OS-9 Assembler/Linker User Manual	1771-6.5.106

Application Program Interface (API) Error Codes

DTL, CC, and PCCC Error Codes

The API error codes are returned by the BPI_, CC_, DTL_, MSG_, and TAG_ functions. The following table lists all API error codes for the control coprocessor and a description of each code. For more information, see the 1771 Control Coprocessor User Manual, publication 1771-6.5.95.

Decimal Value	Hex Value	Symbolic Name	Description
0	0	DTL_SUCCESS or CC_SUCCESS	Successful operation
3	3	DTL_E_DEFBAD2	Invalid number of elements to DEFINE
4	4	DTL_E_DEFBAD3	Invalid data type
5	5	DTL_E_DEFBAD4	Invalid access rights
9	9	DTL_E_DEFBADN	Invalid number of DEFINE parameters
11	B	DTL_E_FULL	DEFINE table full
15	F	DTL_E_R_ONLY	Data item defined as READ_ONLY
16	10	DTL_E_INVTYPE	Data is of invalid type for operation
17	11	DTL_E_NO_MEM	Not enough memory available
18	12	DTL_E_TIME	I/O operation did not complete in time
19	13	DTL_E_NOINIT	DEFINE table not initialized
20	14	DTL_E_BADID	Definition ID out of range
24	18	DTL_E_FAIL	I/O complete with errors
25	19	DTL_E_BADPARAM	Bad parameter value
26	1A	DTL_E_NOPARAM	Expected parameter missing
27	1B	DTL_E_NOATMPT	I/O operation was not attempted
31	1F	DTL_E_TOOBIG	Data item greater than maximum allowed
32	20	DTL_E_NODEF	No such data item defined

API error codes continued . . .

Decimal Value	Hex Value	Symbolic Name	Description
38	26	DTL_E_DFBADADR	Bad DEFINE address
39	27	DTL_E_NOREINIT	DTL system already initialized
40	28	DTL_E_INPTOOLONG	DEFINE input string too long
41	29	DTL_E_CNVT	Data-conversion error
42	2A	DTL_E_GETIME	PLC-5® time invalid
50	32	DTL_E_BADDEF	Invalid use of definition
101	65	CC_E_PENDING	I/O operation in progress
117	75	CC_E_NO_MEM	Not enough memory available
118	76	CC_E_TIME	Operation did not complete in time
120	78	CC_E_BADID	Define ID out of range
124	7C	CC_E_FAIL	I/O completed with errors
127	7F	CC_E_NOATMPT	I/O operation not attempted
131	83	CC_E_TOOBIG	Data size greater than maximum allowed
133	85	CC_E_BAD_MSGID	Message ID out of range (0–31)
141	8D	CC_E_CNVT	Data-conversion error
157	9D	CC_E_NOTCONNECT	PLC® not connected or offline
159	9F	CC_E_NOEXPANDER	Expander not present
160	A0	CC_E_INV_TO	Invalid timeout value
161	A1	CC_E_INV_PORT	Invalid port address
162	A2	CC_E_INV_CTYPE	Invalid coprocessor data type
163	A3	CC_E_INV_PTYPE	Invalid PLC-5 data type
164	A4	CC_E_INV_BPI_MASK	Invalid value for BPI trigger mask
165	A5	CC_E_BAD_RACK	Rack value out of range

API error codes continued . . .

Decimal Value	Hex Value	Symbolic Name	Description
166	A6	CC_E_BAD_GROUP	Group value out of range
167	A7	CC_E_BAD_MODULE	Module slot value out of range
168	A8	CC_E_BAD_RETRY	Retry value out of range
171	AB	CC_E_TIME_LOCKED	Did not complete in time, TAG locked
172	AC	CC_E_TIME_NOREAD	Did not complete in time, TAG not read
173	AD	CC_E_TIME_NOWRITE	Did not complete in time, TAG not written
174	AE	CC_E_TAGPEND	TAG WRITE already pending on this TAG
175	AF	CC_E_BADTAG	Invalid TAG name
176	B0	CC_E_NOTAG	TAG not found
177	B1	CC_E_TAGFULL	TAG table full
178	B2	CC_E_NOTLOCKER	Caller not locker of this TAG
179	B3	CC_E_NOTDEFINER	Caller not definer of this TAG
181	B5	CC_E_MSGPEND	Message already pending
182	B6	CC_E_MSG_ABORT	Message aborted by CC_MKILL
183	B7	CC_E_MSGNOTPEND	Message not pending
184	B8	CC_E_DUP	Duplicate TAG
185	B9	CC_E_NOTLOCKED	TAG not locked
186	BA	CC_E_BADACC	Bad value for TAG access
189	BD	CC_E_NOACCESS	TAG READ_ONLY
190	BE	CC_E_SIZE	Invalid size for operation
191	BF	CC_E_TOOSMALL	Size of buffer too small
192	C0	CC_E_INVTYPE	Invalid type for operation

API error codes continued . . .

Decimal Value	Hex Value	Symbolic Name	Description
258	102	PCCC_E_102	Remote station did not acknowledge command
259	103	PCCC_E_103	Duplicate token holder detected on link
260	104	PCCC_E_104	Channel disconnected from link
261	105	PCCC_E_105	Timed out waiting for a response from remote station
262	106	PCCC_E_106	Duplicate node address detected on link
263	107	PCCC_E_107	Communication channel offline or inactive
264	108	PCCC_E_108	Hardware fault on communication channel
272	110	PCCC_E_110	Illegal command or format, including odd address
288	120	PCCC_E_120	Host has problem and will not communicate
304	130	PCCC_E_130	Remote station host not there, disconnected, or shut down
320	140	PCCC_E_140	Host could not complete function due to hardware fault
336	150	PCCC_E_150	Addressing problem or memory protect rungs
352	160	PCCC_E_160	Function disallowed due to command protection selection
368	170	PCCC_E_170	Processor in program mode
384	180	PCCC_E_180	Compatibility mode file missing or communication zone
400	190	PCCC_E_190	Remote station cannot buffer command
432	1B0	PCCC_E_1B0	Remote station problem due to download
448	1C0	PCCC_E_1C0	Cannot execute command due to IBPs

API error codes continued . . .

Decimal Value	Hex Value	Symbolic Name	Description
513	201	PCCC_E_201	Illegal address format; a field has an illegal value
514	202	PCCC_E_202	Illegal address format; not enough fields specified
515	203	PCCC_E_203	Illegal address format; too many fields specified
516	204	PCCC_E_204	Illegal address; symbol not found
517	205	PCCC_E_205	Illegal address; symbol 0 or greater than 8 characters
518	206	PCCC_E_206	Illegal address; address does not exist
519	207	PCCC_E_207	Illegal size
520	208	PCCC_E_208	Cannot complete request; situation changed
521	209	PCCC_E_209	Data too large
522	20A	PCCC_E_20A	Size too big
523	20B	PCCC_E_20B	No access, privilege violation
524	20C	PCCC_E_20C	Resource not available
525	20D	PCCC_E_20D	Resource already available
526	20E	PCCC_E_20E	Command cannot be executed
527	20F	PCCC_E_20F	Overflow; histogram overflow
528	210	PCCC_E_210	No access
529	211	PCCC_E_211	Incorrect type data
530	212	PCCC_E_212	Bad parameter
531	213	PCCC_E_213	Address reference exists to deleted area
532	214	PCCC_E_214	Command execution failure for unknown reason

API error codes continued . . .

Decimal Value	Hex Value	Symbolic Name	Description
533	215	PCCC_E_215	Data-conversion error
536	218	PCCC_E_218	1771 module response was not valid size, checksum, etc.
534	216	PCCC_E_216	1771 rack adapter not responding
535	217	PCCC_E_217	Timed out, 1771 backplane module not responding

Decimal Value	Hex Value	Symbolic Name	Description
537	219	PCCC_E_219	Duplicated label
538	21A	PCCC_E_21A	File open—another station owns it
539	21B	PCCC_E_21B	Another station is program owner

BASIC Function Codes

The BASIC function codes are used to call API functions from a BASIC program; they appear as the first argument to the AB_BAS call. The following table lists the BASIC function codes and a description of each code. For more information, see the 1771 Control Coprocessor User Manual, publication 1771-6.5.95.

Function Name	Function Number	Description
CC_INIT	0	Initializes internal data structures and installs trap handler
DTL_INIT	1	Creates and initializes DTL data-definition table
DTL_C_DEFINE	2	Adds a definition to DTL data-definition table
DTL_UNDEF	3	Deletes a data definition from DTL data-definition table
DTL_DEF_AVAIL	4	Returns number of data definitions that can be added to DTL data-definition table
DTL_READ_W	5	Reads data from PLC-5 programmable-controller data table to control-coprocessor memory
DTL_WRITE_W	6	Writes data from control-coprocessor memory to PLC-5 programmable-controller data table
DTL_RMW_W	7	Initiates an operation that: reads a data element; modifies some of bits; then writes it back
DTL_GET_WORD	8	Gets a word from a byte array
DTL_GET_FLT	9	Gets a floating-point value from a byte array
DTL_GET_3BCD	10	Gets a 3-digit BCD value from a byte array
DTL_GET_4BCD	11	Gets a 4-digit BCD value from a byte array
DTL_PUT_WORD	12	Puts a word into raw format
DTL_PUT_FLT	13	Puts a floating-point value into a byte array—you can use this array to write to a data item whose PLC data type is FLOAT and whose control-coprocessor data type is RAW
DTL_PUT_3BCD	14	Puts a 3-digit BCD value into a byte array
DTL_PUT_4BCD	15	Puts a 4-digit BCD value into a byte array

BASIC function codes continued . . .

Function Name	Function Number	Description
DTL_SIZE	16	Gets size of memory necessary to store contents of a data item in control-coprocessor format
DTL_TYPE	17	Gets control-coprocessor data type of named data
DTL_CLOCK	18	Sets control-coprocessor date and time to same date and time found in PLC-5 programmable controller
DTL_READ_W_IDX	20	Reads any elements of a file, one element at a time, from PLC-5 programmable controller to control-coprocessor memory
DTL_WRITE_W_IDX	21	Writes any elements of a file, one element at a time, from control-coprocessor memory to PLC-5 programmable controller
DTL_RMW_W_IDX	22	Initiates an operation that reads any element of a PLC-5 processor file, modifies some of the bits based on mask values, then writes the data element back
BPI_DISCRETE	32	Gets updated output-image word; optionally, sets input-image word
BPI_WRITE	33	Responds to a synchronous Block Transfer Read from a programmable controller

Function Name	Function Number	Description
BPI_READ	34	Responds to a synchronous Block Transfer Write from a programmable controller
MSG_READ_W_HANDLER	40	Handles a PLC-5 programmable-controller generated message-read instruction
MSG_READ_HANDLER	41	Handles a PLC-5 programmable-controller message-read instruction
MSG_WRITE_W_HANDLER	42	Handles a PLC-5 programmable-controller generated message-write instruction
MSG_WRITE_HANDLER	43	Handles a PLC-5 programmable-controller message-write instruction
MSG_CLR_MASK	44	Clears bit associated with specified message number
MSG_SET_MASK	45	Sets bit associated with specified message number
MSG_TST_MASK	46	Tests bit associated with specified message number

BASIC function codes continued . . .

Function Name	Function Number	Description
MSG_ZERO_MASK	47	Zeros all bits in specified mask
MSG_WAIT	48	Waits for one or more messages to complete
TAG_DEFINE	60	Adds an entry to control-coprocessor TAG table
TAG_UNDEF	61	Removes a TAG or TAGs from TAG table defined by calling process
TAG_GLOBAL_UNDEF	62	Removes a TAG or TAGs from TAG table defined by any calling process
TAG_DEF_AVAIL	63	Returns number of TAG definitions available in TAG table
TAG_LINK	64	Gets handle from TAG name
TAG_READ_W	65	Reads data from a user's TAG memory area after TAG is written by TAG_WRITE_W
TAG_READ	66	Reads data from a user's TAG memory area
TAG_WRITE_W	67	Writes data to a user's TAG memory area then waits for it to be read by a TAG_READ_W
TAG_WRITE	68	Writes data to a user's TAG memory area
TAG_LOCK	69	Locks requested TAG memory area
TAG_UNLOCK	70	Unlocks requested TAG memory area
CC_ERROR	100	Gets a pointer to a NULL-terminated "canned" error message
CC_ERRSTR	101	Copies "canned" NULL-terminated error message into user's local buffer
CC_DISPLAY_STR	102	Copies four characters to ASCII display
CC_GET_DISPLAY_STR	103	Copies characters of current ASCII display to user's buffer
CC_DISPLAY_HEX	104	Displays an unsigned integer value in 3-character hexadecimal (with trailing "H") on ASCII display

BASIC function codes continued . . .

Function Name	Function Number	Description
CC_DISPLAY_EHEX	105	Displays an unsigned integer value in 4-character hexadecimal on ASCII display of serial expander module
CC_DISPLAY_DEC	106	Displays an integer value in decimal on ASCII display of serial expander module
CC_PLC_SYNC	107	Synchronizes with PLC-5 program scan (this function can be used with a direct-connect mode control coprocessor only)
CC_PLC_STATUS	108	Returns current status of processor status flags and major fault words (this function can be used with a direct-connect mode control coprocessor only)
CC_STATUS	111	Returns current status information of coprocessor
CC_EXPANDED_STATUS	112	Gets current expanded status information of coprocessor
CC_PLC_BTW	113	Requests PLC-5 programmable controller to perform a Block Transfer Write to an intelligent I/O module
CC_PLC_BTR	114	Requests PLC-5 programmable controller to perform a Block Transfer Read from an intelligent I/O module

OS-9 Error Codes

The OS-9 error codes are returned by OS-9 system calls (in C and BASIC) or during an exit from a command. There are five error code ranges listed and described in the following table.

Range	Description
000:001–000:067	Miscellaneous errors
000:102–000:163	Processor-exception errors—Error codes in this range are reserved to indicate that a processor-related exception occurred on behalf of the program; only those listed within this range can occur on behalf of the user program; all other numbers between 100-163 are reserved Unless the program provides for special handling of the exception condition (F$STrap), the error is fatal and the program terminates The listed errors that fall between 100-163 represent the hardware exception vector plus 100
000:164–000:176	Miscellaneous errors
000:200–000:239	Operating-system errors—these errors are normally generated by the kernel or file managers
000:240–000:255	I/O errors—these error codes are generated by device drivers or file managers

The following table lists OS-9 error codes in numerical order and a description of each code. For more information, see the OS-9 Operating System User Manual, publication 1771-6.5.102.

Error Number	Description
000:001	Process has aborted
000:002	KEYBOARD QUIT—Keyboard abort signal (S$Abort) was sent; this is usually generated by typing <control> <E>
000:003	KEYBOARD INTERRUPT—Keyboard interrupt signal (S$Intrpt) was sent; this is usually generated by typing <control> <C>
000:004	MODEM HANGUP—Modem hangup signal (S$HangUp) was sent; this is usually generated when device driver detects loss of data carrier

OS-9 error codes continued . . .

Error Number	Description
000:064 E$IllFnc	ILLEGAL FUNCTION CODE—A math trap handler error
000:065 E$FmtErr	FORMAT ERROR—A math trap handler error
000:066 E$NotNum	NUMBER NOT FOUND—A math trap handler error
000:067 E$IllArg	ILLEGAL ARGUMENT—A math trap handler error
000:102 E$BusErr	BUS ERROR—A bus error exception occurred
000:103 E$AdrErr	ADDRESS ERROR—An address error exception occurred
000:104 E$IllIns	ILLEGAL INSTRUCTION—Illegal instruction exception occurred
000:105 E$ZerDiv	ZERO DIVIDE—An integer zero divide exception occurred
000:106 E$Chk	CHECK—A CHK or CHK2 instruction exception occurred
000:107 E$TrapV	TRAP—A TRAPV, TRAPcc, or FTRAPcc instruction exception occurred
000:108 E$Violat	PRIVILEGE VIOLATION—A privilege violation exception occurred
000:109 E$Trace	UNINITIALIZED TRACE EXCEPTION—An uninitialized trace exception occurred
000:110 E$1010	1010 TRAP—An A Line emulator exception occurred
000:111 E$1111	1111 TRAP—An F Line emulator exception occurred
000:113	COPROCESSOR PROTOCOL VIOLATION
000:114	FORMAT ERROR
000:115	UNINITIALIZED INTERRUPT OCCURRED
000:124	SPURIOUS INTERRUPT OCCURRED
000:133–000:147 E$Trap	Uninitialized user TRAP 1-15 executed
000:148 E$FPUnordC	FPCP ERROR—Branch or set on unordered condition error
000:149 E$FPInxact	FPCP ERROR—Inexact result
000:150 E$FPDivZer	FPCP ERROR—Divide by zero error
000:151 E$FPUndrFl	FPCP ERROR—Underflow error
000:152 E$FPOprErr	FPCP ERROR—Operand error
000:153 E$FPOverFl	FPCP ERROR—Overflow error
000:154 E$FPNotNum	FPCP ERROR—Not a number signaled
000:155	FPCP ERROR—Unimplemented data type
000:156	PMMU CONFIGURATION ERROR
000:157	PMMU ILLEGAL OPERATION
000:158	PMMU ACCESS LEVEL VIOLATION
000:164 E$Permit	NO PERMISSION—Process or module must be owned by super-user to perform requested function
000:165 E$Differ	DIFFERENT ARGUMENTS—F$ChkNam arguments do not match

OS-9 error codes continued . . .

Error Number		Description
000:166	E$StkOvf	STACK OVERFLOW—F$ChkNam can cause this error if pattern string is too complex
000:167	E$EvntID	ILLEGAL EVENT ID—An invalid or illegal event ID number is specified
000:168	E$EvNF	EVENT NAME NOT FOUND—An attempt to link to or delete an event is made, but name is not found in event table
000:169	E$EvBusy	EVENT BUSY—An attempt to delete an event is made and its link count is non-zero; this can also occur if an attempt to create an already existent named event is made
000:170	E$EvParm	IMPOSSIBLE EVENT PARAMETER—This error returns when impossible parameters are passed to F$Event
000:171	E$Damage	SYSTEM DAMAGE—A system data structure has been corrupted
000:172	E$BadRev	INCOMPATIBLE REVISION—Software revision is incompatible with operating-system revision
000:173	E$PthLost	PATH LOST—Path became lost; this usually occurs when a network node has gone down, a serial connection has lost data carrier, or a pipe path has been broken due to an SS_Brreak SetStat
000:174	E$BadPart	BAD PARTITION—Bad partition data or no active partition
000:175	E$Hardware	HARDWARE DAMAGE HAS BEEN DETECTED—E$Hardware usually occurs when driver fails to detect correct responses from hardware; this can occur due to hardware failure or an incorrect hardware configuration
000:176	E$SectSize	INVALID SECTOR SIZE—Sector size of a RBF device must be a binary multiple of 256 (256, 512, 1024, etc.); maximum sector size is 32768
000:200	E$BPNum	PATH TABLE FULL—A user program has tried to open more than 32 I/O paths simultaneously; this error could be returned if there is not enough contiguous memory to expand table

OS-9 error codes continued . . .

Error Number	Description	Error Number	Description
000:201 E$BPNum	ILLEGAL PATH NUMBER—Path number was too large or for a non-existent path; this could occur whenever passing a path number to an I/O call	000:207 E$MemFul	MEMORY FULL—Process will not execute because there is not enough contiguous RAM free; this can also occur if a process has already been allocated maximum number of blocks permitted by system
000:202 E$Poll	INTERRUPT POLLING TABLE FULL—An attempt was made to install an IRQ Service Routine into system polling table, and table was full; to install another interrupt producing device, one must first be removed	000:208 E$UnkSvc	ILLEGAL SERVICE REQUEST—Specified service call has an unknown or invalid service code number; this can also occur if a Getstat/ Setstat call is made with an unknown status code
000:203 E$BMode	ILLEGAL MODE—An attempt was made to perform an I/O function of which device or file was incapable; this could occur, for instance, when trying to read from an output file (for example, a printer)	000:209 E$ModBsy	MODULE BUSY—An attempt was made to access a non-sharable module that is in use by another process
000:204 E$DevOvf	DEVICE TABLE FULL—Specified device cannot be added to system because device table is full; to install another device, one must first be removed	000:210 E$BPAddr	BOUNDARY ERROR—A memory deallocation request was not passed a valid block address or an attempt was made to deallocate memory not previously assigned
000:205 E$BMID	ILLEGAL MODULE HEADER—Specified module cannot be loaded because its module sync code is incorrect	000:211 E$EOF	END OF FILE—An end-of-file condition was encountered on a read operation
000:206 E$DirFul	MODULE DIRECTORY FULL—Specified module cannot be added to system because module directory is full; to load or create another module, one must first be unlinked; this error may be returned because there is not enough memory or memory is too fragmented to use	000:212 E$VctBsy	VECTOR BUSY—A device is trying to use an IRQ vector that is currently being used by another device

OS-9 error codes continued . . .

Error Number		Description
000:213	E$NES	NON-EXISTING SEGMENT—A search was made for a disk file segment that cannot be found; device may have a damaged file structure
000:214	E$FNA	FILE NOT ACCESSIBLE—An attempt was made to open a file or device without correct access permissions; check file's attributes and owner ID
000:215	E$BPNam	BAD PATH NAME—There is a syntax error in specified pathlist (illegal character, etc.); this can occur whenever referencing a path by name
000:216	E$PNNF	PATH NAME NOT FOUND—Specified pathlist cannot be found; this could be caused by misspellings or incorrect directories, etc.
000:217	E$SLF	SEGMENT LIST FULL—A file is too fragmented to be expanded any further; this can be caused by expanding a file many times without regard to allocation of memory; it also occurs on disks with little free memory or disks whose free memory is too scattered
000:218	E$CEF	FILE ALREADY EXISTS—An attempt was made to create a file using a name that already appears in current directory
000:219	E$IBA	ILLEGAL BLOCK ADDRESS—A search for an illegal block address has occurred; an invalid pointer or block size has been passed or device's file structure is damaged
000:220	E$HangUp	TELEPHONE (MODEM) DATA CARRIER LOST
000:221	E$MNF	MODULE NOT FOUND—A request is made to link to a module that is not found in module directory; modules whose headers have been modified or corrupted will not be found
000:222	E$NoClk	NO CLOCK—This error returns when a request is made that uses system clock and system has no clock running; SETIME is used to start system clock
000:223	E$DelSP	SUICIDE ATTEMPT—A user requested deallocation and return of memory where user's stack is located; this could be caused, for example, by using F$Mem system call to contract data memory of specified process
000:224	E$IPrcID	ILLEGAL PROCESS NUMBER—A system call was passed a process ID to a non-existent process or a process that user may not access
000:225	E$Param	BAD PARAMETER—A service request has been passed an illegal or impossible parameter
000:226	E$NoChld	NO CHILDREN—An F$Wait request was made and process has no child process for which to wait

OS-9 error codes continued . . .

Error Number		Description
000:227	E$ITrap	ILLEGAL TRAP CODE—An unavailable (already in use) or invalid trap code is used in a TLINK call
000:228	E$PrcAbt	PROCESS ABORTED—A process is aborted by kill signal code
000:229	E$PrcFul	PROCESS TABLE FULL—System process table is full (too many processes currently running); although OS-9 automatically tries to expand table, this error may occur if there is not enough contiguous memory to do so
000:230	E$IForkP	ILLEGAL PARAMETER AREA—Ridiculous parameters were passed to a fork call
000:231	E$KwnMod	KNOWN MODULE—A call was made to install a module that is already in memory
000:232	E$BMCRC	INCORRECT MODULE CRC—Specified module being checked or verified has a bad CRC value; to generate a valid CRC, use FIXMOD utility
000:233	E$USigP	UNPROCESSED SIGNAL PENDING
000:234	E$NEMod	NON-EXECUTABLE MODULE—A process tries to execute a module with a type other than program/object
000:235	E$BNam	BAD NAME—There is a syntax error in specified name
000:236	E$BMHP	BAD PARITY—Specified module has bad module header parity

Error Number		Description
000:237	E$NoRAM	RAM FULL—There is no free system RAM available at time of request for memory allocation; this also occurs when there is not enough contiguous memory to process a fork request
000:238	E$DNE	DIRECTORY NOT EMPTY—Attempt made to remove directory attribute from a directory that is not empty
000:239	E$NoTask	NO TASK NUMBER AVAILABLE—All task numbers are currently in use and a request was made for execution or creation of a new task
000:240	E$Unit	ILLEGAL DRIVE NUMBER
000:241	E$Sect	BAD SECTOR—Bad disk sector number
000:242	E$WP	WRITE PROTECT—Device is write protected
000:243	E$CRC	CRC ERROR—CRC error on read or write verify
000:244	E$Read	READ ERROR—Data transfer error during disk read operation, or SCF (terminal) input buffer overrun
000:245	E$Write	WRITE ERROR—Hardware error during disk write operation
000:246	E$NotRdy	NOT READY—Device has “not ready” status

OS-9 error codes continued . . .

Error Number	Description	Error Number	Description
000:247 E$Seek	SEEK ERROR—Physical seek to non-existent sector	000:252 E$Lock	RECORD IS LOCKED-OUT—Another process is accessing requested record; normal record locking routines will wait forever for a record in use by another user to become available; however, RBF may be told to wait for a finite amount of time with a Setstat; if time expires before record becomes free, this error returns
000:248 E$Full	MEDIA FULL—Insufficient free space on media	000:253 E$Share	NON-SHARABLE FILE BUSY—Requested file or device has single user bit set or it was opened in single user mode and another process is accessing requested file
000:249 E$BTyp	WRONG TYPE—Attempt to read incompatible media (e.g., attempt to read double-sided disk on single-sided drive)	000:254 E$DeadLk	I/O DEADLOCK—Two processes are attempting to use same two disk areas simultaneously; each process is locking out other process, producing I/O deadlock; one of two processes must release its control to allow other to proceed
000:250 E$DevBsy	DEVICE BUSY—Non-sharable device is in use	000:255 E$Format	DEVICE IS FORMAT PROTECTED—An attempt was made to format a disk that is format protected; a bit in device descriptor may be changed to allow device to be formatted; formatting is usually inhibited on hard disks to prevent erasure
000:251 E$DIDC	DISK ID CHANGE—Disk media was changed with open files; RBF copies disk ID number (from sector 0) into path descriptor of each path when it is opened; ff this does not agree with driver's current disk ID, this error returns		

C-Compiler Error Messages

The C-compiler error messages are issued by the C compiler (`cc` or `xcc`) during preprocessing (page 3-7) or compiling (page 3-1). See “Assembler/Linker Error Messages” on page 4-1 for messages issued by the assembler (`r68`) or linker (`l68`). There are four types of error messages:

- Warning (W)
- Error (E)
- Fatal Error (F)
- Compiler Error (E)

C-Compiler Error Messages

The following table lists the C-compiler generated error messages and a description of each error message. Each error message listing identifies the message type. For more information, see the OS-9 C Language User Manual, publication 1771-6.5.104.

Error Message	Description
already a local variable (E)	Variable has already been declared at current block level
argument error (E)	Function argument declared as type function; pointers to functions are allowed
argument storage (E)	Function arguments may only be declared as storage class register
bad character (E)	A character not in C character set (probably a control character) was encountered in source file
both must be integral (E)	>> and << operands cannot be FLOAT or DOUBLE
break error (E)	Break statement is allowed only inside a while, do, for or switch
can’t determine size (E)	Size of object cast cannot be determined
can’t initialize unions (E)	Unions cannot appear as object of an initializer
can’t open strings file (E)	A temporary file for constant string storage could not be opened; likely cause is no permission in directory or no room left on device
can’t take address (E)	& operator not allowed on a register variable; operand must otherwise be an lvalue

C-compiler error messages continued . . .

Error Message	Description	Error Message	Description
can’t take size of bitfield (E)	Sizeof operator cannot be applied to bit fields	condition needed (E)	While, do, for, switch and if statements require a condition expression
cannot cast as function or array (E)	Type result of cast cannot be FUNCTION or ARRAY	constant expression required (E)	Initializer expressions for static or external variables cannot reference variables; they may, however, refer to address of a previously declared variable This installation allows no initializer expressions unless all operands are of type INT or CHAR; variables are not allowed for array dimensions or cases
cannot evaluate size (E)	Could not determine size from declaration or initializer	constant operator (C)	Internal compiler error[1]
cannot initialize (E)	Storage class or type does not allow variable to be initialized	constant overflow (E)	Input numeric constant was too large for implied or explicit type
cannot use void (E)	No operation can be performed on a void	continue error (E)	Continue statement is allowed only inside a while, do, or for block.
case value too large for type (C)	Internal compiler error[1]	declaration mismatch (E)	This declaration conflicts with a previous one; this is typically caused by declaring a function to return a non-integer type after a reference has been made to function
compiler tag validation (C)	Internal compiler error[1]	degenerative comparison with zero (W)	A comparison of form u>=0 or u<0, where u is unsigned, is being done.
compiler trouble (C)	Compiler detected something it couldn’t handle; try compiling program again; if this error still occurs, contact Microware	deref (C)	Internal compiler error[1]

[1] If this message is reproducible, try to isolate the problem and submit a Product Discrepancy Report to Microware.

C-compiler error messages continued . . .

Error Message	Description
deref storage (C)	Internal compiler error[1]
dimension mismatch (E)	An array has been declared twice, with conflicting bounds
divide by zero (E)	Divide by zero occurred when evaluating a constant expression
dumpstrings (F)	An error occurred during processing of strings file; likely cause is no more room on output device
duplicate cases (E)	All constant values used as a switch statement case must be unique
duplicate member name (E)	A member identification name in a struct/union declaration has already appeared in this struct/union
duplicate struct/union tags (E)	TAG name of this struct/union has already been defined at current block level
error writing assembly code file (E)	An error occurred when writing output file; likely cause is no room left on output device
? expected (E)	? is any character that was expected to appear here; missing semicolons or braces cause this error
expression missing (E)	An expression is required here
expression too complex (C)	This expression could not be compiled by compiler; if simplifying expression (using temporaries) does not help, contact Microware
expression with little effect (W)	This expression calculates a value that is never used
function header missing (E)	Statement or expression encountered outside a function; typically caused by mismatched braces
function type error (E)	A function cannot be declared as returning an array, function, struct, or union
function unfinished (E)	End-of-file encountered before end of function definition
gen: unk opr (C)	Internal compiler error[1]
identifier missing (E)	Identifier name required here but none was found
identifier name found in a cast (E)	Identifier name found in a cast; only a cast types are allowed
illegal declaration (E)	Declarations are allowed only at beginning of a block
illegal pointer/integer combination (W)	Mixing pointer and integer types may result in non-portable code

[1] If this message is reproducible, try to isolate the problem and submit a Product Discrepancy Report to Microware.

C-compiler error messages continued . . .

Error Message	Description
illegal type combination (E)	Operators for indicated operator do not have compatible types
input line too long (F)	Source input line that was read is too long; maximum length of a source input line is 512 characters
invalid enumeration constant value (E)	Explicit values for enumerated type constants must be constant integral expressions
Invalid bit field type (E)	Only int and unsigned are permissible types for bit fields
invalid bit field size (E)	Width of a bit field must be a constant integral expression between 1 and 32
label ’<label>’ undefined (E)	Goto label not defined in current function
label ’<label>’ unused (W)	Named label was defined but never referenced
label required (E)	Goto statement requires a label identifier as an operand
lvalue required (E)	Left side of assignment must be able to be *stored into*; array names, functions, etc. are not lvalues
multiple defaults (E)	Only one default statement is allowed in a switch block

Error Message	Description
multiple definition (E)	Identifier name was declared more than once in same block level
must be integral (E)	Type of object required here must be type int, char, or pointer
named twice (E)	Names in a function parameter list may appear only once
no ’if’ for ’else’ (E)	Else statement found with no matching if; this is typically caused by extra or missing braces and/or semicolons
no switch statement (E)	Case statements can only appear within a switch block
not a function (E)	Primary in expression is not type “function returning...” if this is really a function call, function name was declared differently elsewhere
not a member of this strict/union (W)	Identifier given as member name is not a member of declared struct/union aggregate type
not an argument (E)	Name does not appear in function parameter list
operand expected (E)	Unary operators require one operand, binary operators two; this is typically caused by misplaced parentheses, casts, or operators
operands have incompatible types (E)	Operands for indicated operator do not have compatible types

Error Message	Description
out of memory (F)	Compiler dynamic memory overflow; compiler requires dynamic memory for symbol table entries, block level declarations, and code generation
pointer mismatch (E)	Pointers refer to different types; use a cast if required
pointer or integer required (E)	A pointer (of any type) or integer is required to left of –> operator
pointer required (E)	Pointer operand required with unary * operator
pointer type mismatch (W)	Mixing pointer types may not allow portability of code
possible degenerate assignment in test (W)	This assignment in a conditional assignment in test expression may actually be a bug
primary expected (E)	Primary expression required here
reg free (C)	Internal compiler error[1]
rel op (C)	Internal compiler error[1]
’return;’ in non-void function (W)	A non-value returning return statement was found in a function returning a value of type other than void
return value type mismatch (W)	Type of expression returned by a return statement does not match type of declared function
should be NULL (E)	Second and third expression of ?: conditional operator cannot be pointers to different types; if both are pointers, they must be of same type or one of two must be null
**** STACK OVERFLOW **** (F)	Compiler stack has overflowed; most likely cause is very deep block-level nesting
storage error (E)	Reg and auto storage classes may only be used within functions
struct syntax, expecting brace (E)	Brace, comma, etc. is missing in a struct declaration
struct or union inappropriate (E)	Struct or union cannot be used in this context

[1] If this message is reproducible, try to isolate the problem and submit a Product Discrepancy Report to Microware.

C-compiler error messages continued . . .

Error Message	Description
struct/union size exceeds 32k (E)	Total size of a struct or union cannot exceed 32767 bytes
struct/union member required (E)	Identifier on right side of arrow (–>) or period (.) operator must be a struct/union member identifier name
struct/union object required (E)	Primary expression (left side) of period (.) operator must be a struct/union object
struct/union pointer mismatch (W)	Mixing struct/union pointer types may not allow portability of code
struct/union pointer required (E)	Primary expression (left side) of arrow (–>) operator must be a struct/union pointer
struct/union type is not allowed (E)	A struct or union type is not allowed as an operand for given operation
syntax error (E)	Expression, declaration, or statement is incorrectly formed
syntax misplaced arg declaration list (E)	An argument declaration list is improperly placed in a function declarator
third expression missing (E)	A question mark (?) must be followed by a colon (:) with expression; this error may be caused by unmatched parentheses or other errors in expression
too long (E)	Too many characters provided in a string initializing a character array
too many braces (E)	Unmatched or unexpected braces encountered processing an initializer
too many elements (E)	More data items supplied for aggregate level in initializer than members of aggregate
type expected (E)	A type name was expected but not found
typedef not a variable (E)	Typedef type name cannot be used in this manner
undeclared identifier (E)	No declaration exists at any block level for this identifier
undefined structure (E)	Union or struct declaration refers to an undefined structure name
undefined struct/union tag referenced (E)	A struct or union object was referenced TAG but has not yet been defined
uregfree (C)	Internal compiler error[1]

[1] If this message is reproducible, try to isolate the problem and submit a Product Discrepancy Report to Microware.

C-compiler error messages continued . . .

Error Message	Description
unterminated character constant (E)	Unmatched ’ (character delimiters)
unterminated string (E)	Unmatched “ (string delimiters)
while expected (E)	No while found for do statement

Preprocessor Error Messages

The following table lists the C-compiler preprocessor-generated error messages and a description of each error message. Each error message identifies the message type. For more information, see OS-9 C Language User Manual, publication 1771-6.5.104.

Error Message	Description
#if nesting too deep (E)	Maximum nesting for #if/#ifdef directives is 32 levels
error writing output file (F)	Error occurred writing output file—commonly caused by running out of space on output storage device
illegal #if macro name (E)	An illegal identifier was found in a #if/#ifdef directive
illegal '#' (E)	An illegal directive was found on a pound sign (#) preprocessor line
illegal macro name (E)	An illegal identifier was found during macro definition
incorrect include file (E)	File name given in an #include directive must be delimited by double quotes (“ ”) or angle brackets (< >)
macro arguments required (E)	This macro was defined with arguments, but none were given when called
macro definition error (E)	A syntax error was found during a macro definition; macro dummy arguments must be a list of valid identifiers enclosed in parentheses; white space is required after defining parenthesis
missing #endif (E)	End of current file was reached and a pending #if/#ifdef or #else was in effect
no #if for #else (E)	An #else directive was encountered without a corresponding #if/#ifdef
out of memory (E)	No more memory is available to continue processing
redefined macro (W)	Indicated macro name has already been defined; use #undef <name> (carefully) if it is intended to redefine a macro name
source file read error (F)	An error occurred reading input source file
source file too long (F)	Maximum length of an input line is 512 characters
too few macro arguments (E)	A macro was called without enough arguments to match macro definition
too many #endifs (E)	An #endif directive was encountered before a corresponding #if/#ifdef
too many macro arguments (E)	A macro was called with more arguments than given in macro definition

Assembler/Linker Error Messages

The assembler/linker error messages are issued by the `r68` assembler (page 4-1) or the `l68` linker (page 4-3), and they can appear during the assembly or linkage phases when assembling an assembly-language program or when compiling a C program.

Assembler Error Messages

These error messages consist of brief phrases that describe the kind of error that the assembler detected. The following table lists each error message and a description of each message. For more information, see the OS-9 Assembler/Linker User Manual, publication 1771-6.5.106.

Error Message	Description
Bad label	Statement's label has an illegal character or does not begin with a letter
Bad Mnemonic	A mnemonic was found in mnemonic field that was not recognized or was not allowed in current program section
Bad number	Numeric constant definition contains a character that is not allowed in current radix
Bad operand	An operand expression is missing or incorrectly formed
Bad operator	An arithmetic expression is incorrectly formed
Bad option	An option is unrecognized or incorrectly specified
Bracket missing	Opening or closing bracket is missing
Can't open file	A problem was encountered opening an input file
Can't open macro work file	A problem was encountered opening a macro work file
Comma expected	A comma was expected but not found
Conditional nesting error	A mismatched if/else/endc conditional assembly directive was found
Constant definition	A constant definition is incorrectly formed
ENDM without MACRO	An endm was found with no matching macro
ENDR without REPT	An endr was found with no matching rept
Fail <message>	A fail directive was encountered
File close error	A problem was encountered closing an input file

Assembler error messages continued . . .

Error Message	Description
Illegal addressing mode	Addressing mode cannot be used in instruction
Illegal external reference	External names cannot be used with assembler directives; if an operand expression contains an external name, only operation allowed in expression is binary plus and minus
Illegal index register	Register cannot be used as an index register
Illegal suffix	An illegal suffix was found in an instruction
Label missing	This statement is missing required label
Macro arg too long	Macro argument is too long; no more than 60 characters total can be passed to a macro
Macro file error	A problem was encountered accessing macro work file
Macro nesting too deep	Macro calls are nested too deeply; macro calls may only be nested up to 8 levels deep
Nested MACRO definitions	Macro cannot be defined inside a macro definition
Nested REPT	Repeat blocks cannot be nested
New symbol in pass two	See symbol lost
No input files	An input file must be specified
No param for arg	A macro expansion is attempting to access an argument that was not passed by macro call
Phasing error	A label has a different value during pass two than it did during pass one
Redefined name	Name appears more than once in label field other than on a set directive
Symbol lost?	Assembler symbol lookup error; error could be caused by symbol table overflow or bad memory
Too many args	Too many arguments were passed to macro; no more than 9 arguments may be passed to a macro
Too many object files	Only one –o= command line option is allowed
Undefined org“*”	(program counter org) can’t be accessed within a vsect
Unmatched quotes	A beginning or ending quotation mark was expected but not found

Linker Error Messages

The following table lists linker error, warning, and information messages issued by the OS-9/68000 linker (`l68`) and a description of each message. In this table, the following syntax conventions are used:

- <file> represents the actual file name in question
- <n> represents the actual number in question
- <char> represents the actual character in question

Error Message	Description
<file> contains a 6809 module	A module from 6809 assembler was encountered
<file> contains assembly errors	A module was encountered that had assembly errors; fix errors and re-link
<file> contains no root psect	First file given on command line must contain a root psect—a root psect is psect from which all references are resolved; it is specified by non-zero type and language fields in module's psect directive
<file> created by assembler too new for this linker	The `r68` and `l68` programs are not compatible editions; be sure correct programs are installed in execution directory
<file> is not a relocatable module	Relocatable module header in '<file>' was either not present or incorrectly formed; all relocatable object headers start with bytes: $DE $AD $FA $CE; use dump utility on input file to verify this error
<file> rof<4 and code>32k. Must be re-assembled.	This message is caused when linker processes an old version of assembler output that contains more than 32k of code; re-assembly of source file will fix problem
bad syscrc size	This is an internal linker error[1]
can't create output file	Output file for module (given by `-o=` option) cannot be created; possible causes are no access permissions or no disk space
can't create symbol file	Symbol file for module cannot be created; possible causes are no access permissions or no disk space
can't open <file>	One of input files given could not be opened; possible causes are no access permissions, non-existent file, or no free memory

[1] If you can reproduce this error at will, contact Microware.

Linker error messages continued . . .

Error Message	Description
can't open <file> name file	The –z=<file> could not be opened; possible causes are no access permissions, non-existent file, or no free memory
can't reopen input file <file>	This is an internal linker error[1]
duplicate symbol names	Linker has determined that same symbol name appears in more than one psect in allocation of final module
error reading input file <file>	Linker could not read input file; either a physical error occurred or input file was incorrectly formatted; all input files must be output from assembler
error writing file <file>	Linker could not write output file; possible causes are disk errors or media full
initialized data (or gullable) allowed only on program or trap handler modules	Initialized data is supported only for program modules (entered by F$Fork) or trap handler modules (entered by F$TLink); modules such as system modules, device drivers, and file managers cannot have initialized data
jmp total > guess (<n>/<n>)	This is an internal linker error[1]
no data storage allocation (vsect) allowed on non-object modules	Only modules of type object code can contain data storage allocation; other module types (usually language runtime interpreters) define data storage allocations in a different manner
no initialized static data allowed on raw output	Initialized data cannot be allocated to a program designed to run without OS-9; uninitialized data is allowed; linker prints size of uninitialized data requirement for raw modules
no root psect found	First module given on command line must contain a root psect—a root psect is a module in which psect directive indicates a non-zero type and language word; a zero type and language word means a subroutine module
non-remote data allocation value exceeds 64k	See discussion in manual on data area references for a description of linker memory limits
odd count for crc	This is an internal linker error[1]

[1] If you can reproduce this error at will, contact Microware.

Linker error messages continued . . .

Error Message	Description
operand size error	This message occurs when an operand value exceeds legal range for size of operand
out of memory	Linker cannot obtain enough memory to do linkage; memory usage requirements depend on many factors: number of input files, number of psects, number of global symbols and undefined references; a psect that is 128k long requires a 128k buffer to link
reference location error (<n>)	An internal linker error has occurred; this error is caused by information from assembler that linker does not expect; if this happens, be sure assembler and linker are properly installed on system from original distribution medium[1]
root psect found in both <file1> and <file2>	Only one root psect is allowed for a program—a root psect is defined as a psect in which Type/ Language field is non-zero; root psect is initial psect from which all external references are resolved
symbol ‘name’ not found during pass 2	This is an internal linker error[1]
too many data references <n>	This is an internal linker error[1]
unknown option –<char>	An option given is not an option recognized by linker
unknown reference type <n>	This is an internal linker error[1]
unresolved references	Symbols previously listed by linker are not defined by a psect given on command lines or in libraries; this is commonly caused by improperly ordered library files

[1] If you can reproduce this error at will, contact Microware.

BASIC Error Codes

The BASIC error codes are used in BASIC and RUNB for run-time errors or for errors in BASIC statement syntax detected at program entry. The following table lists BASIC error codes and a description for each code. Error codes above 80 are those used by OS-9 or other external programs. For more information, see the OS-9 BASIC User Manual, publication 1771-6.5.103.

Error Message	Description
10	Unrecognized Symbol
11	Excessive Verbage (too many keywords or symbols)
12	Illegal Statement Construction
13	I-code Overflow (need more workspace memory)
14	Illegal Channel Reference (bad path number given)
15	Illegal Mode (Read/Write/Update/Dir only)
16	Illegal Number
17	Illegal Prefix
18	Illegal Operand
19	Illegal Operator
20	Illegal Record Field Name
21	Illegal Dimension
22	Illegal Literal
23	Illegal Relational
24	Illegal Type Suffix
25	Too-Large Dimension
26	Too-Large Line Number
27	Missing Assignment Statement
28	Missing Path Number
29	Missing Comma
30	Missing Dimension
31	Missing DO Statement
32	Memory Full (need more workspace memory)
33	Missing GOTO
34	Missing Left Parenthesis
35	Missing Line Reference
36	Missing Operand
37	Missing Right Parenthesis
38	Missing THEN statement
39	Missing TO
40	Missing Variable Reference
41	No Ending Quote

BASIC error codes continued . . .

Error Message	Description	Error Message	Description
42	Too Many Subscripts	62	I/O Format Repeat Error
43	Unknown Procedure	63	I/O Format Syntax Error
44	Multiply-Defined Procedure	64	Illegal Path Number
45	Divide by Zero	65	Wrong Number of Subscripts
46	Operand Type Mismatch	66	Non-Record-Type Operand
47	String Stack Overflow	67	Illegal Argument
48	Unimplemented Routine	68	Illegal Control Structure
49	Undefined Variable	69	Unmatched Control Structure
50	Floating Overflow	70	Illegal FOR Variable
51	Line with Compiler Error	71	Illegal Expression Type
52	Value out of Range for Destination	72	Illegal Declarative Statement
53	Subroutine Stack Overflow	73	Array Size Overflow
54	Subroutine Stack Underflow	74	Undefined Line Number
55	Subscript out of Range	75	Multiply-Defined Line Number
56	Parameter Error	76	Multiply-Defined Variable
57	System Stack Overflow	77	Illegal Input Variable
58	I/O Type Mismatch	78	Seek Out of Range
59	I/O Numeric Input Format Bad	79	Missing Data Statement
60	I/O Conversion: Number out of Range	80	Print Buffer Overflow
61	Illegal Input Format		

Internet Error Codes

The following table lists error messages and a description of each message that can be returned by socket access to the Internet software. Some of these error codes are appropriate for the FTP and TELNET programs. Some error codes are never returned in this implementation but are listed here for future compatibility. For more information, see the OS-9 Internet Software Reference Manual, publication 1771-6.4.11.

Error Message	Description
007:001	I/O operation would block—an operation that would cause a process to block was attempted on a socket in non-blocking mode
007:002	I/O operation now in progress—an operation that takes a long time to complete (such as connect()) was attempted on a socket in non-blocking mode
007:003	Operation already in progress—an operation was attempted on a non-blocking object that already had an operation in progress
007:004	Destination address required—attempted socket operation requires a destination address.
007:005	Message too long—a message sent on a socket was larger than internal message buffer or some other network limit; messages should be smaller than 32768 bytes
007:006	Protocol wrong type for socket—a protocol was specified that does not support semantics of socket type requested; for example, an AF_INET UDP protocol as SOCK_STREAM is wrong protocol type for socket
007:007	Bad protocol option; a bad option or level was specified in getsockopt() or setsockopt()

Error Message	Description
007:008	Protocol not supported—requested protocol is not available or not configured for use
007:009	Socket type not supported—requested socket type is not supported or not configured for use
007:010	Operation not supported on socket—for example, accept() on a datagram socket
007:011	Protocol family not supported
007:012	Address family not supported by protocol
007:013	Address already in use—only one use of each address is normally permitted; wildcard use and connectionless communication are exceptions
007:014	Can’t assign requested address—normally results from an attempt to create a socket with an address not on this machine

Internet error codes continued . . .

Error Message	Description
007:015	Network is down—network hardware is not accessible
007:016	Network is unreachable; usually caused by network interface hardware that is operational but not physically connected to network; this error can also be caused when network has no way to reach destination address
007:017	Network dropped connection on reset; host you were connected to crashed and rebooted
007:018	Software caused connection abort; a connection abort was caused by local (host) machine
007:019	Connection reset by peer—a peer forcibly closed a connection; this normally results from a loss of connection on remote socket due to a time out or reboot
007:020	No buffer space available—a socket operation could not be performed because system lacked sufficient buffer space or a queue was full
007:021	Socket is already connected—a connect() request was made on an already connected socket; also caused by a sendto() request on a connected socket to a destination that is already connected
007:022	Socket is not connected—a request to send or receive data was rejected because socket was not connected or no destination was given with a datagram socket
007:023	Can’t send after socket shutdown
007:024	Too many references
007:025	Connection timed out—a connect() or send() request failed because connected peer did not properly respond after a period of time; time out period depends on protocol used
007:026	Connection refused by target—no connection could be established because target machine actively refused it; this usually results from trying to connect to a service that is inactive on target host
007:027	Mbuf too small for mbuf operation
007:028	Socket module already attached
007:029	Path is not a socket

A-B QUALITY
ALLEN-BRADLEY
A ROCKWELL INTERNATIONAL COMPANY

Allen-Bradley has been helping its customers improve productivity and quality for 90 years. A-B designs, manufactures and supports a broad range of control and automation products worldwide. They include logic processors, power and motion control devices, man-machine interfaces and sensors. Allen-Bradley is a subsidiary of Rockwell International, one of the world's leading technology companies.

With major offices worldwide.

Algeria • Argentina • Australia • Austria • Bahrain • Belgium • Brazil • Bulgaria • Canada • Chile • China, PRC • Colombia • Costa Rica • Croatia • Cyprus • Czech Republic • Denmark Ecuador • Egypt • El Salvador • Finland • France • Germany • Greece • Guatemala • Honduras • Hong Kong • Hungary • Iceland • India • Indonesia • Israel • Italy • Jamaica • Japan Jordan • Korea • Kuwait • Lebanon • Malaysia • Mexico • New Zealand • Norway • Oman • Pakistan • Peru • Philippines • Poland • Portugal • Puerto Rico • Qatar • Romania Russia–CIS • Saudi Arabia • Singapore • Slovakia • Slovenia • South Africa, Republic • Spain • Switzerland • Taiwan • Thailand • The Netherlands • Turkey • United Arab Emirates United Kingdom • United States • Uruguay • Venezuela • Yugoslavia

World Headquarters, Allen-Bradley, 1201 South Second Street, Milwaukee, WI 53204 USA, Tel: (1) 414 382-2000 Fax: (1) 414 382-4444

Publication 1771-7.1—December 1994
Supersedes Publication 1771-7.1—May 1993

PN 955119-25