PCBoard Programming Language

Reference Guide

(Extracted from Power PPL 2.0 HLP File)

Thanks to By Francis Gastellu, developer os Power PPL, that included the entire reference in Power PPL HLP

PPL Reference Index

Compiler Options

There are two options for the compiler :

1 - Array Dimension Checking:

This option allow the compiler to check for mistakes when using the statement REDIM. for example :

```
DIM TABLE(5,5,5)
...
REDIM TABLE,10,10
```

This will generate an error unless Array Dimension Checking has been disabled.

2 - User Variable Generation

This allow the PPE to generate the user variables when starting... disable this option if you don't want to have the work done... Usually, there is no need to change anything here... both options may be checked unless you have reasons to changed that.

ASCII Table

The ASCII code

American Standard Code for Information Interchange

www.theasciicode.com.ar

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

theASCIIcode.com.ar

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ò	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

PCBoard Programming Language Reference Index

Functions & Statements - A

Abort
Abs
Account
ActMsgNum
AdjBytes
AdjDBytes
AdjTBytes
AdjTFiles
AdjTime
Alias
And
AnsiOn
AnsiPos
Append
Asc

Function & Statements - B

B2w
Backup
BitClear
BitSet
Blt
Break
Broadcast
Bye

Function & Statements - C

Call
CallID
CallNum
Carrier
CcType
CdCheckOff
CdCheckOn
CdOn
Chat
ChatStat
Chr
CloseCap
ClrEol
Cls
Color
ConfAlias
ConfExp
ConfFlag
ConfMw
ConfReg
ConfSel
ConfSys
ConfUnFlag

Continue
Copy
Crc32
CurColor
CurConf
CurSec
CurUser

Function & Statements - D

Date
Day
DbgLevel
Dec
Declare
DefAns
DefColor
Delay
Delete
DelUser
Dir
DispFile
DispStr
DispText
DNext
DoIntr
Dow
Download
DriveSpace
DtrOff
DtrOn

Function & Statements - E

End
ErrorCorrect
EvtTimeAdj
Exist

Function & Statements - F

FAppend
FClose
FCloseAll
FCreate
FDefIn
FDefOut
FDGet
FDPut
FDPutLn
FDPutPad
FDRead
FDWrite
Ferr
FFlush

FGet
FileInf
Flag
FlagCnt
FmtCC
FmtReal
FNext
FOpen
ForNext
Forward
FPut
FPutLn
FPutPad
FRead
FReAltUser
FreshLine
FRewind
FSeek
Function
FWrite

Function & Statements - G

GetAltUser
GetEnv
GetToken
GetUser
GetX
GetY
Go
Goodbye
GoSub
GoTo
GrafMode

Function & Statements - H

Hangup
HelpPath
HiConfNum
HiMsgNum
Hour

Function & Statements - I

I2s
IfThen
InBytes
Inc
Inkey
Input
InputCC
InputDate
InputInt
InputMoney

InputStr
InputText
InputTime
InputYN
Instr
IsBitSet
IsNonStop

Function & Statements - J

Join

Function & Statements - K

KbdBufSize
KbdChkOff
KbdChkOn
KbdFile
KbdFileUsed
KbdFlush
KbdString
KbdStuff
KeyFlush
KInkey

Function & Statements - L

Lang
LangExt
LastAns
LastIn
Left
Len
Let
Log
LoggedOn
LoMsgNum
Loop
Lower
Lprinted
Ltrim

Function & Statements - M

Mask_Alnum
Mask_Alpha
Mask_Ascii
Mask_File
Mask_Num
Mask_Path
Mask_Pwd
MaxNode
MdmFlush
MegaNum
Message
MGetByte

Mid
Min
MInkey
MinLeft
MinOn
Mixed
MkAddr
MkDate
Modem
Month
More
MouseReg
MPrint
MPrintLn
MsgToFile

Function & Statements - N

NewLine
NewLines
NoChar
Not

Function & Statements - O

OnLocal
OpenCap
Operators
OpText
Or
OutBytes

Function & Statements - P

PageOff
PageOn
PageStat
PCBAccount
PcbAccStat
PcbDat
PcbMac
PcbNode
PeekB
PeekDW
PeekW
PokeB
PokeDW
PokeW
Pop
PPEName
PPEPath
PPLBufSize
PRFound
Print
PrintLn

Procedure
PromptStr
Psa
Push
PutAltUser
PutUser

Function & Statements - Q

Quest
Quit
QwkLimits

Function & Statements - R

Random
RdUnet
RdUsys
ReadLine
RecordUsage
ReDim
RegAh
RegAl
RegAx
RegBh
RegBl
RegBx
RegCf
RegCh
RegCl
RegCx
RegDh
RegDi
RegDl
RegDs
RegDx
RegEs
RegF
RegSi
Rename
Replace
ReplaceStr
ResetDisp
RestScrn
Return
Right
Rtrim

Function & Statements - S

S2i
SaveScrn
ScanMsgHdr
ScrFile
ScrText

SearchFind
SearchInit
SearchStop
Sec
Select Case
SendModem
SetEnv
SetLmr
Shell
ShowOff
ShowOn
ShowStat
SlPath
Sort
Sound
Space
SPrint
SPrintLn
StackAbort
StackErr
StackLeft
StartDisp
Stop
String
Strip
StripAtx
StripStr
Syntax
SysopSec

Function & Statements - T

TempPath
Time
TimeAP
To
ToDDate
TokCount
Tokenize
TokenStr
ToType
TPACGet
TPACPut
TPACRead
TPACWrite
TPAGet
TPAPut
TPARead
TPAWrite
Trim

Function & Statements - U

Un_City

Un_Name
Un_Oper
Un_Stat
Upper
UserAlias
U_Bdl
U_BdlDay
U_Bul
U_Fdl
U_Ful
U_InConf
U_LDate
U_LDir
U_Lmr
U_Logons
U_LTime
U_MsgRd
U_MsgWr
U_Name
U_PwdHist
U_PwdLc
U_PwdTc
U_RecNum
U_Stat
U_TimeOn

Function & Statements - V

ValCC
ValDate
ValTime
VarAddr
VarOff
VarSeg
Ver

Function & Statements - W

Wait
WaitFor
While
WrUnet
WrUsys
WrUsysDoor

Function & Statements - X

Xor

Function & Statements - Y

Year
YesChar

PPL SOURCE SYNTAX

Each line of a PPL source file may contain none, one, some or all of the following sections:

[KEYWORD][EXPR|VAR][,EXPR|VAR][;|'][COMMENT TEXT]

KEYWORD - A PPL statement used to accomplish some task.
EXPR - A PPL expression which may contain VARs, CONSTs, and/or FUNCs.
VAR - A PPL variable with optional array subscript.
CONST - A PPL constant.
FUNC - A PPL function that returns a value.
; - Used to start a comment. Ignored by the compiler.
' - Used to start a comment. Ignored by the compiler.
* - Used to start a comment if first character of the line.
COMMENT - Comment text following the ; or '. Ignored by the compiler.

If a line is blank or contains only a comment, it is skipped. if it contains a KEYWORD, that line is compiled into a tokenized format. If it doesn't contain a KEYWORD but some argument, it is assumed to be an assignement statement (LET).
A double quote ("") may be embedded within a string constant to tell the compiler that a single literal quote is desired In other words, "THIS""IS""A""TEST" would evaluate to THIS"IS"A"TEST after the leading and trailing quotes are removed and the double quotes were folded to single quotes.
Labels and variable names may now include the following characters in addition to A-Z, 0-9, and the _ (underscore) character: $ (dollar sign), @ (commercial at), # (pound sign), ¢ (cents), £ (british pound), ¥ (japanese yen)
A \ (backslash) character as the last character on a line (before any comments) will now allow continuing a logical line from one to the next physical line
A : (colon) character may be used to separate multiple logical lines on a single physical line

DATA TYPES

SYNTAX

TYPE var[(dim[,dim[,dim]])][,var...]

PPL utilizes the following data types:

BOOLEAN

unsigned character (1 byte) 0 = FALSE, non-0 = TRUE

DATE

unsigned integer (2 bytes) PCBoard julian date (count of days since 1/1/1900)

DDATE

Signed long integer for julian date. DDATE is for use with DBase date fields. It holds a long integer for julian dates. When coerced to string type it is in the format CCYYMMDD or 19940527

INTEGER / SDWORD / LONG

signed long integer (4 bytes) Range: -2,147,483,648 -> +2,147,483,647

MONEY

signed long integer (4 bytes) Range: -$21,474,836.48 -> +$21,474,836.47

STRING

far character pointer (4 bytes) NULL is an empty string non-NULL points to a string of some length less than or equal to 256

TIME

signed long integer (4 bytes) Count of seconds since midnight

BIGSTR

Allows up to 2048 characters per big string (up from 256 for STRING variables) May include CHR(0) characters in the middle of the big string (unlike STRING variables which may not)

EDATE

Julian date in earth date format Deals with dates formatted YYMM.DD Range: Same as DATE

REAL / FLOAT

4-byte floating point number Range: +/-3.4E-38 - +/-3.4E+38 (7-digit precision)

DREAL / DOUBLE

8-byte floating point number Range: +/-1.7E-308 - +/-1.7E+308 (15-digit precision)

UNSIGNED / DWORD / UDWORD

4-byte unsigned integer Range: 0 - 4,294,967,295

BYTE / UBYTE

1-byte unsigned integer Range: 0 - 255

WORD / UWORD

2-byte unsigned integer Range: 0 - 65,535

SBYTE / SHORT

1-byte signed integer Range: -128 - 127

SWORD / INT

2-byte signed integer Range: -32,768 - 32,767

NOTES

Any type may be assigned to any other type. This is the simplest way to accomplish type conversion. BOOLEAN, DATE, INTEGER, MONEY and TIME are all integer types and may be assigned to each other. Assignment from a larger data type to a smaller data type automatically converts the data to a format suitable for the smaller data type. Conversion to and from STRINGs is dependent on the other data type. DATEs are imported/exported as "MM-DD-YY". TIMEs are imported/exported as "HH:MM:SS". MONEYs are imported/exported as "#.##" without embedded dollar signs or commas, and using as many characters as needed to the left of the decimal point. All variables must be declared before use.

CONSTANTS

PPL allows user defined constants in the following formats:

$#.##

A MONEY constant (dollar sign followed by optional dollars followed by decimal point followed by cents; # = 0-9)
□□□
#h

An INTEGER hexadecimal constant (# = 0-9 & A-F)
□□□
#d

An INTEGER decimal constant (# = 0-9)
□□□
#o

An INTEGER octal constant (# = 0-7)
□□□
#b

An INTEGER binary constant (# = 0-1)
□□□
#

An INTEGER constant (# = 0-9)
□□□
"X"

A STRING constant (X = any displayable text)
□□□
  @X##

An INTEGER @X constant (# = 0-9 & A-F)

PREDEFINED CONSTANTS

PPL predefines the following constants:

  AUTO = 2000h

Parameter passed to INPUTSTR and PROMPTSTR statements (automatically press enter after 10 seconds of no user input)

  BELL = 800h

Parameter passed to DISPTEXT statement (sound a bell when prompt displayed)
□□□□
  CRC_FILE - CRC_STR

These constants were added to avoid confusion when telling the function CRC32 what it is taking the CRC of. CRC_FILE tells CRC32 to calculate the CRC of the file contained within the string argument. CRC_STR tells CRC32 to calculate the CRC of the string argument itself. CRC_FILE has a value of 1 (TRUE) CRC_STR has a value of 0 (FALSE)

  CUR_USER = 0

Parameter passed to CURUSER()

  DEFS = 0

Parameter passed to various statements for default values

  ECHODOTS = 1h

Parameter passed to INPUTSTR and PROMPTSTR statements (echo dots instead of user input)

  ERASELINE = 20h

Parameter passed to INPUTSTR and PROMPTSTR statements (erase the current line when user presses enter)

  FALSE = 0

BOOLEAN FALSE value

  FCL = 2

Value passed to STARTDISP to force line counting display

FIELDLEN = 2h

Parameter passed to INPUTSTR and PROMPTSTR statements (displays parenthesis to show input field width if ANSI enabled) □□□□

FNS = 1

Value passed to STARTDISP to force non-stop display

F_EXP = 2h

Expired subscription access allowed flag for CONFFLAG and CONFUNFLAG

F_MW = 10h

Mail waiting flag for CONFFLAG and CONFUNFLAG

F_REG = 1h

Registered access allowed flag for CONFFLAG and CONFUNFLAG

F_SEL = 4h

Conference selected flag for CONFFLAG and CONFUNFLAG

F_SYS = 8h

Conference SysOp access flag for CONFFLAG and CONFUNFLAG

GRAPH = 1h

Parameter passed to DISPFILE statement to search for graphics specific files

GUIDE = 4h

Parameter passed to INPUTSTR and PROMPTSTR statements (displays parenthesis above current line if FIELDLEN used and ANSI not enabled) □□□□

HIGHASCII = 1000h

Parameter passed to INPUTSTR and PROMPTSTR statements (allow high ascii characters, regardless of current valid character set, if disable high ascii filter set to yes)

LANG = 4h

Parameter passed to DISPFILE statement to search for language specific files

LFAFTER = 100h

Parameter passed to INPUTSTR, PROMPTSTR and DISPTEXT statements (send an extra line feed after user presses enter)

LFBEFORE = 80h

Parameter passed to INPUTSTR, PROMPTSTR and DISPTEXT statements (send an extra line feed before prompt display)

LOGIT = 8000h

Parameter passed to DISPTEXT statement (log text to callers log)

LOGITLEFT = 10000h

Parameter passed to DISPTEXT statement (log text to callers log, forcing left justification)

NC = 0

Value passed to STARTDISP to not change display mode

NEWLINE = 40h

Parameter passed to INPUTSTR, PROMPTSTR and DISPTEXT statements (send a line feed after user presses enter)

NOCLEAR = 400h

Parameter passed to INPUTSTR and PROMPTSTR statements (don't clear field at first keypress regardless of ANSI)

NO_USER = -1

Parameter passed to CURUSER()

O_RD = 0

Parameter passed to FCREATE/FOPEN/FAPPEND to open a file in read only mode

O_RW = 2

Parameter passed to FCREATE/FOPEN/FAPPEND to open a file in read and write mode

O_WR = 1

Parameter passed to FCREATE/FOPEN/FAPPEND to open a file in write only mode

SEC = 2h

Parameter passed to DISPFILE statement to search for security specific files

STACKED = 10h

Parameter passed to INPUTSTR and PROMPTSTR statements (allow semi-colons and spaces in addition to valid character set passed)

S_DB = 3h

Parameter passed to FCREATE/FOPEN/FAPPEND to deny read and write (both) access from other processes

S_DN = 0h

Parameter passed to FCREATE/FOPEN/FAPPEND to allow read and write (deny none) access from other processes

S_DR = 1h

Parameter passed to FCREATE/FOPEN/FAPPEND to deny read access from other processes

S_DW = 2h

Parameter passed to FCREATE/FOPEN/FAPPEND to deny write access from other processes

TRUE = 1

BOOLEAN TRUE value

UPCASE = 8h

Parameter passed to INPUTSTR and PROMPTSTR statements (force user input to upper case)

WORDWRAP = 200h

Parameter passed to INPUTSTR and PROMPTSTR statements (if user hits end of line, save the text at the end of the line for future use)

YESNO = 4000h

Parameter passed to INPUTSTR and PROMPTSTR statements (Only allow international yes/no responses)

NO_USER = -1

Return by GetUser - variables are currently undefined

CUR_USER = 0

Return by GetUser - User variables are for the current user
See also : Predefined Variables

PREDEFINED VARIABLES

PPL predefines the following variables for user record access:

BOOLEAN U_CLS

Clear screen between messages status

BOOLEAN U_DEF79

79 column message editor default

BOOLEAN U_EXPERT

Users current expert status

BOOLEAN U_FSE

Users full screen editor default

BOOLEAN U_FSEP

Prompt for full screen editor status

BOOLEAN U_LONGHDR

6 line vs 4 line message header status

BOOLEAN U_SCROLL

Scroll multi-screen message status

DATE U_EXPDATE

The users subscription expiration date

DATE U_PWDEXP

The date that the users password expires and must be changed

INTEGER U_EXPSEC

The users expired security level

INTEGER U_PAGELEN

The users page length

INTEGER U_SEC

The users security level

STK_LIMIT

This constant was added so the PPL programmer could determine how close they are getting to the stack limit when using recursion.

STRING U_ADDR(5)

The users address information (if the SysOp has enabled address recording)

Subscript 0 = First street line
1 = Second street line
2 = City
3 = State
4 = Zip
5 = Country

STRING U_ALIAS

The users alias (if the SysOp has enabled alias use)

STRING U_BDPHONE

The users business/data phone number

STRING U_CITY

The users city/state information

STRING U_CMNT1

The users comment field

STRING U_CMNT2

The SysOps comment field

STRING U_HVPHONE

The users home/voice phone number

STRING U_NOTES(4)

Notes about the user (if the SysOp has enabled the note capability)
Subscripts 0-4 hold lines 1-5

STRING U_PWD

The users password

STRING U_TRANS

The users default transfer protocol

STRING U_VER

The users verification string (if the SysOp has enabled user verification)
See also : Predefined Constants
□□

EXPRESSION OPERATORS

PPL allows the following operators to be used in expressions (lvalue and rvalue are simply the values to the left and right, respectively, of binary operators):

```
(      - Start sub-expression (also allows [ to be used)
)      - End sub-expression (also allows ] to be used)
^      - Raise lvalue to the power of rvalue (also allows ** to be used)
*      - Multiply lvalue by rvalue
/      - Divide lvalue by rvalue
%      - Remainder of lvalue divided by rvalue
+      - Add rvalue to lvalue
-      - Subtract rvalue from lvalue
=      - Is lvalue equal to rvalue (also allows ==)
<>     - Is lvalue not equal to rvalue (also allows != and ><)
<      - Is lvalue less than rvalue
<=     - Is lvalue less than or equal to rvalue (also allows =<)
>      - Is lvalue greater than rvalue
>=     - Is lvalue greater than or equal to rvalue (also allows =>)
!      - Logical not of rvalue
&      - Logical and of lvalue with rvalue (also allows &&)
|      - Logical or of lvalue with rvalue (also allows ||)
```

ABORT() :BOOLEAN

Returns a flag indicating whether or not the user aborted the display of data via ^K/^X or answering no to a MORE? Prompt

ABS(var:integer) :INTEGER

Returns the absolute value of "var"

AND(var1:integer,var2:integer) :INTEGER

Returns the bitwise and of two integer expressions

See also : Or Xor Not

ANSION() :BOOLEAN

Returns TRUE if the user has ANSI capabilities
See also : OnLocal GrafMode

ASC(var:string) :INTEGER

Returns the ASCII value (0-255) of the first character of "var"
See also : Chr

B2W(var1:integer,var2:integer) :INTEGER

Returns a word built from two byte sized values by the formula:
(var1*0100h+var2)

CALLID() :STRING

Returns the caller ID string

CALLNUM() :INTEGER

Returns the caller number of the current user.

CARRIER() :INTEGER

Returns the carrier speed as reported by the modem to PCBoard
See also : ErrCorrect

CCTYPE(var:string) :STRING

Returns the issuer of credit card number "var"
See also : FmtCC InputCC ValCC

CDON() :BOOLEAN

Returns TRUE if the carrier detect signal is on
See also : CdCheckOn CdCheckOff

CHATSTAT() :BOOLEAN

Return the current users chat availability status (TRUE means available, FALSE means unavailable).
See also : PageStat

CHR(var:integer) :BIGSTR

Returns a single character long string of the character represented by ASCII code "var" (0-255)
See also : Asc

CONFREG(confNum:integer) :BOOLEAN

Returns TRUE if users registered flag is set, FALSE otherwise
See also : ConfSel ConfSys ConfMw CurConf ConfExp CurConf
ConfFlag ConfUnFlag Join ConfAlias LastIn

CONFEXP(confNum:integer)

Returns TRUE if users expired flag is set, FALSE otherwise
NOTE:
ConfReg() = FALSE & CONFEXP = TRUE, user locked out ConfReg() = TRUE &
CONFEXP = TRUE, user reg & exp
See also : ConfSel ConfSys ConfMw CurConf CurConf
ConfFlag ConfUnFlag Join ConfAlias LastIn
ConfReg

CONFSEL(confNum:integer) :BOOLEAN

Returns TRUE if user has selected the conference, FALSE otherwise
See also : ConfSys ConfMwCurConf ConfExp CurConf
ConfFlag ConfUnFlag Join ConfAlias LastIn
ConfReg

CONFSYS(confNum:integer) :BOOLEAN

Returns TRUE if user has conference SysOp access, FALSE otherwise
See also : ConfSel ConfMwCurConf ConfExp CurConf
ConfFlag ConfUnFlag Join ConfAlias LastIn
ConfReg

CONFMW(confNum:integer) :BOOLEAN

Returns TRUE if user has mail waiting in conference "confnum", FALSE otherwise
See also : ConfSel ConfSys CurConf ConfExp CurConf
ConfFlag ConfUnFlag Join ConfAlias LastIn
ConfReg

CURCOLOR() :INTEGER

Returns the current color (0-255) in use by the ANSI driver
See also : DefColor

CURCONF() :INTEGER

Returns the current conference number

See also : ConfSel ConfSys ConfMw ConfExp CurConf
ConfFlag ConfUnFlag Join ConfAlias LastIn
ConfReg

CURSEC() :INTEGER

Returns the users current security level
See also : SysopSec

□□

DATE() :DATE

Returns todays date
See also : Time

DAY(datevar:date) :INTEGER

Returns the day of the month (1-31) of "datevar" □□□□
See also : Month Year Dow

DBGLEVEL() :INTEGER

Returns the debug level in effect

DBGLEVEL dbg:integer

Set the debug level to "dbg"

DEFANS() :BIGSTR

Returns the last default answer passed to an Input statement. For example, this allows a PPE to determine what the default answer would have been had a PCBTEXT prompt not been replaced with a PPE.
See also : LastAns

DEFCOLOR() :INTEGER

Returns the default color as specified in PCBSetup

DEFCOLOR

Resets the current color to the system default
See also : CurColor

DOW(day:date) :INTEGER

Returns the day of the week (0 = Sunday, 6 = Saturday) that "day" fell on □□□□
See also : Date Month Year

ERRCORRECT() :BOOLEAN

Returns TRUE if a session is determined to be error corrected (or FALSE for non-error corrected sessions).
See also : Carrier

EVTTIMEADJ() :BOOLEAN

Detects if the users time has been adjusted for an upcoming event. This is useful to detect if a users time left can be increased with the AdjTime statement.
See also : AdjTime

EXIST(file:string) :BOOLEAN

Returns a boolean TRUE value if the file "file" exists
See also : Delete Copy Append FileInf Rename

FERR(channel:integer) :BOOLEAN

Returns TRUE if a file access error occurred on channel "channel" since the file was opened or FERR was last called
See also : FOpen

□□
FILEINF(file:string,option:integer) :MULTITYPE

Returns a piece of information (specified by "option") about the file "file"
Valid values for "options": 1 = Return TRUE if file exists
2 = Return file date stamp
3 = Return file time stamp
4 = Return file size
5 = Return file attributes
01h = Read Only
02h = Hidden
04h = System
20h = Archive
6 = Return file drive
7 = Return file path
8 = Return file base name
9 = Return file extension

Return value type is depending on the info requested. It may be BOOLEAN, DATE, INTEGER, STRING and TIME
See also : Delete Copy Append Exist Rename

FMTCC(format:strinf) :STRING

Returns a formatted credit card number based on "format" □□□
See also CcType ValCC InputCC

FMTREAL(realExp:real/dreal,fieldWidth:integer,decimalPlaces:integer)

Formats REAL/DREAL values for display purposes.
realExp = A REAL/DREAL floating point expression
fieldWidth = The minimum number of characters to display
decimalPlaces = The number of characters to display to the right of
the decimal point

GETENV(var:string) :STRING

Returns the value of the environment variable named by "var"

GETTOKEN() :STRING

Returns the next string token from a prior call to Tokenize (Same as the GETTOKEN statement but can be used in an expression without prior assignement to a variable)

GETTOKEN VAR

Get a token from a previous call to Tokenize and assign
it to VAR

See also : Tokenize TokenStr TokCount

GETX() :INTEGER

Returns the current column (X position) of the cursor on the display
See also : GetY AnsiPos

GETY() :INTEGER

Returns the current row (Y position) of the cursor on the display
See also : GetX AnsiPos

GRAFMODE() :STRING

Returns a character indicating the users graphics status
R = RIPscrip supported
G = ANSI graphics (color and positioning) supported

A = ANSI positioning (no color) supported
N = No graphics (RIP or ANSI) supported
□□□□
See also : AnsiOn OnLocal
□□

HELPPATH() :STRING
■■■■■■■■■■■■■■■■■■■■

Returns the path, as specified in PCBSetup, to the help files
See also : PPEPath SlPath TempPath

HIMSGNUM() :INTEGER
■■■■■■■■■■■■■■■■■■■■

Returns the high message number for the current conference.
See also : LoMsgNum

HOUR(dayhour:time) :INTEGER
■■■■■■■■■■■■■■■■■■■■■■■■■■■

Returns the hour of the day (0-23) of "dayhour"
See also : Min Sec

I2S(var1:integer,var2:integer) :STRING
■■

Returns a string representing the integer value "var1" converted to base "var2"

See also : S2i String

INKEY() :STRING
■■■■■■■■■■■■■■■■

Returns the next keypress as a single character long string, or a string with the name of the function or cursor control key
See also : KInkey MGetByte MInkey

INSTR(var1:bigstr,var2:bigstr) :INTEGER
■■

Returns the position of "var2" in "var1" (1-LEN(var1)) or 0 if "var2" not in "var1"

ISBITSET(var:multitype, bit:integer) :BOOLEAN
■■■

Check the status of a specified bit in a variable.

This function is primarily intended to be used with BIGSTR variables which can be up to 2048 bytes long. However, it will work with other data types (and expressions) as well if desired.
□□□□

See also : BitSet BitClear

ISNONSTOP() :BOOLEAN

Return whether or not the display is currently in non-stop mode (ie, did the user type NS as part of their command line)
See also : StartDisp

KBDBUFSIZE() :INTEGER

Return the number of key presses pending in the KbdString buffer
See also : PPLBufSize KbdFlush KbdStuff KbdFile KbdString KbdFileUsed MdmFlush KeyFlush KbdFlush

KBDFILEUSED() :BOOLEAN

Return TRUE if key presses are being stuffed via a KbdFile statement.
See also : KbdBufSize PPLBufSize KbdFlush KbdStuff KbdFile MdmFlush KeyFlush KbdFlush KbdString

KINKEY() :STRING

Returns the next keypress from the BBS keyboard as a single character long string, or a string with the name of the function or cursor control key
See also : Inkey MInkey MGetByte

LANGEXT() :STRING

Returns the file extension for the users language selection
See also : Lang

LASTANS() :STRING

function to return the last answer accepted by an Input statement.
See also : DefAns

LEFT(var1:string,var2:integer) :BIGSTR

Returns the left-most "var2" characters of "var1"
See also : Right Mid

LEN(var:bigstr) :INTEGER

Returns the length of "var"

LOGGEDON() :BOOLEAN

Returns TRUE if the user has already logged on to the BBS, FALSE otherwise

LOMSGNUM() :INTEGER

Returns the low message number for the current conference.
See also : HiMsgNum
□□
LOWER(var:bigstr) :BIGSTR

Returns a string of "var" with all uppercase characters converted to lowercase characters
See also : Upper Mixed

LPRINTED() :INTEGER

Return the number of lines printed on the display
See also : StartDisp

LTRIM(var1:bigstr,var2:string) :BIGSTR

Returns a string of "var1" with the first character of "var2" trimmed from the left

See also : Rtrim Trim

MASK_ALNUM() :STRING

Returns a valid character mask for input statements of A through Z, a through z, and 0 through 9
See also : Mask_Alpha Mask_Ascii Mask_File Mask_Num Mask_Path Mask_Pwd

MASK_ALPHA() :STRING

Returns a valid character mask for input statements of A through Z and a through z

See also : Mask_Alnum Mask_Ascii Mask_File Mask_Num Mask_Path Mask_Pwd

MASK_ASCII() :STRING

Returns a valid character mask for input statements of space (" ") through tilde ("~")

□□
See also : Mask_Alpha Mask_Alnum Mask_File Mask_Num Mask_Path Mask_Pwd

MASK_FILE() :STRING

Returns a valid character mask for input statements of file names
See also : Mask_Alpha Mask_Ascii Mask_Alnum Mask_Num Mask_Path Mask_Pwd

MASK_NUM() :STRING

Returns a valid character mask for input statements of 0 through 9
See also : Mask_Alpha Mask_Ascii Mask_File Mask_Alnum Mask_Path Mask_Pwd

MASK_PATH() :STRING

Returns a valid character mask for input statements of path names
See also : Mask_Alpha Mask_Ascii Mask_File Mask_Num Mask_Alnum Mask_Pwd

MASK_PWD() :STRING

Returns a valid character mask for input statements of passwords
See also : Mask_Alpha Mask_Ascii Mask_File Mask_Num Mask_Path Mask_Alnum

MAXNODE() :INTEGER

Returns the maximum node possible with the current software (ie, /2 would return 2, /10 would return 10, etc)
See also : PcbNode

MEGANUM(number:integer)

Converts a decimal number (from 0 to 1295) to a hexa-tri-decimal number, or meganum.

MGETBYTE() :INTEGER

Returns the value of the next byte from the modem (0-255) or -1 if there are no bytes waiting for input
See also : Inkey KInkey MInkey

MID(var1:bigstr,var2:integer,var3:integer) :BIGSTR

Returns a string from "var1" starting at the "var2" position of "var1" and containing "var3" characters of "var1"
See also : Right Left

MIN(var1:time) :INTEGER

Returns the minute of the hour (0-59) of "var1"
See also : Hour Sec

MINKEY() :STRING

Returns the next keypress from the remote caller as a single character long string, or a string with the name of the function or cursor control key
See also : Inkey KInkey MGetByte

MINLEFT() :INTEGER

Returns the current callers minutes left to use online
See also : MinOn

MINON() :INTEGER

Returns the current callers minutes online so far this session
See also : MinLeft

MIXED(var1:string)

Converts a string to mixed (or proper name) case
See also : Upper Lower

MKADDR(seg:integer, off:integer) :INTEGER

Returns a segment:offset address as a long integer built from two word sized values by the formula: (var1*00010000h+var2)
See also : VarSeg VarOff VarAddr

MKDATE(year:integer, month:integer, day:integer) :DATE

Returns a date with the year specified by "year" (1900-2079), month specified by "month" (1-12), and day specified by "day" (1-31).
See also : Year Month Day

MODEM() :STRING

Returns the modem connect string as reported by the modem to PCBoard
See also : Carrier

MONTH(var1:date) :INTEGER

Returns the month of the year (1-12) of "var1"
See also : Year Day Dow

NOCHAR() :STRING

Returns the current language no character
See also : YesChar

NOT(var1:integer) :INTEGER

Returns the bitwise complement (all bits inverted) of an integer expression
See also : Or And Xor

ONLOCAL() :BOOLEAN

Returns TRUE if the user is on locally
See also : AnsiOn GrafMode

OR(var1:integer, var2:integer) :INTEGER

Returns the bitwise or of two integer expressions
See also Xor And Not

PAGESTAT() :BOOLEAN

Returns TRUE if the user has paged the SysOp (or PageOn has been issued), FALSE otherwise (or PageOff has been issued)
See also : ChatStat

PCBDAT() :STRING

Returns a string with the path and file name of PCBOARD.DAT

PCBNODE() :INTEGER

Returns the node number
See also : MaxNode

PEEKB(var:integer) :INTEGER

Returns a byte value (0-255) located at memory address "var" (PEEK is a synonym) - □□□
See also : PeekDW PeekW PokeB PokeW PokeDW

PEEKDW(var:integer) :INTEGER

Returns a signed integer value (-2147483648 - +2147483647) located at memory address "var"
See also : PeekB PeekW PokeB PokeW PokeDW

PEEKW(var:integer) :INTEGER

Returns a word value (0-65535) located at memory address "var"
See also : PeekDW PeekB PokeB PokeW PokeDW

PPENAME() :STRING

Returns the name of the currently executing PPE file minus the path and extension

See also : PPEPath

PPEPATH() :STRING

Returns a string with the path (no file name) of the currently executing PPE file

See also : PPEName

PPLBUFSIZE() :INTEGER

Returns the number of key presses pending in the KbdStuff buffer.
See also : KbdBufSize KbdFlush KbdStuff KbdFile KbdString KbdFileUsed MdmFlush KeyFlush KbdFlush

PSA(var:integer) :BOOLEAN

Returns TRUE if the feature specified by "var" is enabled, FALSE if the feature specified by "var" is disabled

Valid values for var:
1 = Alias Support Enabled
2 = Verify Support Enabled
3 = Address Support Enabled
4 = Password Support Enabled
5 = Statistics Support Enabled
6 = Notes Support Enabled

See also : TPAGet

RANDOM(var:integer) :INTEGER

Returns a random number between 0 and "var" inclusive

READLINE(file:string, line:integer) :STRING

Read and return line number "line" from file "file"

REGAH() :INTEGER

Returns the value of the AH register after a DoIntr statement
See also : RegAl RegAx

REGAL() :INTEGER

Returns the value of the AL register after a DoIntr statement
See also : RegAh RegAx

REGAX() :INTEGER

Returns the value of the AX register after a DoIntr statement
See also : RegAh RegAl

REGBH() :INTEGER

Returns the value of the BH register after a DoIntr statement
See also : RegBl RegBx

REGBL() :INTEGER

Returns the value of the BL register after a DoIntr statement
See also : RegBh RegBx

REGBX() :INTEGER

Returns the value of the BX register after a DoIntr statement
See also : RegBh RegBl

REGCF() :BOOLEAN

Returns the state of the carry flag after a DoIntr statement
See also : RegF

REGCH() :INTEGER

Returns the value of the CH register after a DoIntr statement
See also : RegCl RegCx

REGCL() :INTEGER

Returns the value of the CL register after a DoIntr statement
See also : RegCh RegCx

REGCX() :INTEGER

Returns the value of the CX register after a DoIntr statement
See also : RegCh RegCl

REGDH() :INTEGER

Returns the value of the DH register after a DoIntr statement
See also : RegDl RegDx

REGDI() :INTEGER

Returns the value of the DI register after a DoIntr statement
See also : DoIntr

REGDL() :INTEGER

Returns the value of the DL register after a DoIntr statement
See also : RegDh RegDx

REGDS() :INTEGER

Returns the value of the DS register after a DoIntr statement
See also : DoIntr

REGDX() :INTEGER

Returns the value of the DX register after a DoIntr statement
See also : RegDh RegDl

REGES() :INTEGER

Returns the value of the ES register after a DoIntr statement
See also : DoIntr

REGF() :INTEGER

Returns the value of the flags register after a DoIntr statement
See also : RegCf DoIntr

REGSI() :INTEGER

Returns the value of the SI register after a DoIntr statement
See also : DoIntr

REPLACE(str:bigstr, search:string, replace:string) :BIGSTR

Returns a string of "str" with all occurences of the first character of "search" replaced by the first character of "replace"

See also : ReplaceStr

REPLACESTR(str:bigstr, search:string, replace:string) :BIGSTR

It functions just like the Replace function except that a complete sub-string may be specified for both search and replace
str is the string to work on
search is the string to search for
replace is the string to replace search with

See also : Replace

RIGHT(str:bigstr, len:integer) :BIGSTR

Returns the right-most "len" characters of "str"
See also : Left

RTRIM(str1:bigstr, trim:string) :BIGSTR

Returns a string of "str1" with the first character of "trim" trimmed from the right
See also : Ltrim

S2I(str:string, base:integer) :INTEGER

Returns an integer representing the string "str" converted from base "base"
See also : I2s

SCRTEXT(col:integer, row:integer, len:integer, code:boolean) :STRING

Returns a string with the text (and color information in the form of @X codes if "code" is TRUE) from column "col", row "row", and of length "len"
See also : ScrFile

SEC(var:time) :INTEGER

Returns the second of the minute (0-59) of "var"
See also : Hour Min

SHOWSTAT() :BOOLEAN

Returns TRUE if writing to the display is active, FALSE if writing to the display is disabled
See also : ShowOff ShowOn

SLPATH() :STRING

Returns the path, as specified in PCBSetup, to the login security files
See also : HelpPath PPEPath TempPath

SPACE(len:integer) :BIGSTR

Returns a string of spaces "len" characters long

STRING(var:multitype) :STRING

Returns "var" converted to a string
See also : I2s

STRIP(str:bigstr, char:string) :BIGSTR

Returns a string of "str" with all occurrences of the first character of "char" removed
See also : StripAtx StripStr

STRIPATX(str:bigstr) :BIGSTR

Returns a string of "str" with all @X codes removed □□□□
See also : Strip StripStr

STRIPSTR(str:bigstr, search:string) :BIGSTR

Functions just like the Strip function except that a complete sub-string may be specified for search
str is the string to work on
search is the string to search for
□□□□
See also : Strip StripAtx

SYSOPSEC() :INTEGER

Returns the SysOp security defined in PCBOARD.DAT
See also : CurSec

TEMPPATH() :STRING

Returns the path, as specified in PCBSetup, to the temporary work directory
See also : SlPath HelpPath PPEPath

TIME() :TIME

Returns the current time
See also : TimeAP Date

TIMEAP(var:time) :STRING

Returns a string representing the time "var" in civilian format (XX:XX:XX AM)
See also : Time

TOKCOUNT() :INTEGER

Returns the number of tokens available via the GetToken statement and/or function
See also : Tokenize GetToken TokenStr

TOKENSTR() :STRING

Returns a previously tokenized string reconstructed with semi-colons separating the component tokens
See also : Tokenize GetToken TokCount

TOtype(exp)

TOBOOLEAN, TOMONEY, TOSTRING, TOBIGSTR, TOINTEGER, TOUNSIGNED, TOREAL, TODREAL, TOFLOAT, TODOUBLE, TODATE, TOEDATE, TOTIME, TOBYTE, TOWORD, TODWORD, TOUBYTE, TOUWORD, TOUDWORD, TOSBYTE, TOSWORD, TOSDWORD, TOSHORT, TOINT, & TOLONG
Used to force the result of an expression to a specific type
Usage: TOtype(exp) (returns type)
type is the actual type to force (BIGSTR, BOOLEAN, etc.)
exp is an expression of any type
□□□□
See also : S2i I2s String

TRIM(str:bigstr, char:string) :BIGSTR

Returns a string of "str" with the first character of "char" trimmed from both ends
See also : Rtrim Ltrim

UPPER(str:bigstr) :BIGSTR

Returns a string of "str" with all lowercase characters converted to uppercase characters
See also : Lower Mixed

UN_CITY() :STRING

Returns a nodes city from USERNET.XXX after a RdUnet statement
See also : Un_Name Un_Oper Un_Stat

UN_NAME() :STRING

Returns a nodes user name from USERNET.XXX after a RdUnet statement
See also : Un_City Un_Oper Un_Stat

UN_OPER() :STRING

Returns a nodes operation text from USERNET.XXX after a RdUnet statement
See also : Un_City Un_Name Un_Stat

UN_STAT() :STRING

Returns a nodes status from USERNET.XXX after a RdUnet statement
See also : Un_City Un_Name Un_Oper

U_BDL() :INTEGER

Returns the current users number of bytes downloaded
See also : U_BdlDay U_Bul U_FdlU_Ful U_InConf
U_LDate U_LDir U_LmrU_Logons U_LTime

U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_BDLDAY() :INTEGER
[redacted]

Returns the current users number of bytes downloaded today
See also : U_BdlU_Bul U_FdlU_Ful U_InConf
U_LDate U_LDir U_LmrU_Logons U_LTime
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_BUL() :INTEGER
[redacted]

Returns the current users number of bytes uploaded
See also : U_BdlU_BdlDay U_FdlU_Ful U_InConf
U_LDate U_LDir U_LmrU_Logons U_LTime
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_FDL() :INTEGER
[redacted]

Returns the current users number of files downloaded
See also : U_BdlU_BdlDay U_BulU_Ful U_InConf
U_LDate U_LDir U_LmrU_Logons U_LTime
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_FUL() :INTEGER
[redacted]

Returns the current users number of files uploaded
See also : U_BdlU_BdlDay U_BulU_Fdl U_InConf
U_LDate U_LDir U_LmrU_Logons U_LTime
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_INCONF(record:integer, conf:integer) :BOOLEAN
[redacted]

Returns TRUE if user record number "record" is registered in conference "conf"
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_LDate U_LDir U_LmrU_Logons U_LTime
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_LDATE() :DATE
[redacted]

Returns the current users last date on the system
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDir U_LmrU_Logons U_LTime
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_LDIR() :DATE
[redacted]

Returns the current users last directory scan date
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LmrU_Logons U_LTime
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_LMR(confNum:integer) :INTEGER

function to return the number of the last message read for the specified conference.
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Logons U_LTime
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_LOGONS() :INTEGER

Returns the current users number of times logged on
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Lmr U_LTime
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_LTIME() :TIME

Returns the current users last time on the system
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Lmr U_Logons
U_MsgRd U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_MSGRD() :INTEGER

Returns the number of messages the user has read
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Lmr U_Logons
U_LTime U_MsgWr U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_MSGWR() :INTEGER

Returns the number of messages the user has written
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Lmr U_Logons
U_LTime U_MsgRd U_Name U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_NAME() :STRING

Returns the current users name
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Lmr U_Logons
U_LTime U_MsgRd U_MsgWr U_PwdHistU_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_PWDHIST(hist:integer) :STRING

Returns the specified password from the password history Valid values for "hist" are 1 through 3

See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Lmr U_Logons
U_LTime U_MsgRd U_MsgWr U_Name U_PwdLc
U_PwdTc U_RecNum U_Stat U_TimeOn

U_PWDLC() :DATE

Returns the date of the last password change

See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Lmr U_Logons
U_LTime U_MsgRd U_MsgWr U_Name U_PwdHist
U_PwdTc U_RecNum U_Stat U_TimeOn

U_PWDTC() :INTEGER

Returns the number of times the password has been changed

See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Lmr U_Logons
U_LTime U_MsgRd U_MsgWr U_Name U_PwdHist
U_PwdLc U_RecNum U_Stat U_TimeOn

U_RECNUM(user:string) :INTEGER

Returns the user record number (0-65535) for user name "user" or -1 if "user" is not registered on this system.

See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful
U_InConf U_LDate U_LDir U_Lmr U_Logons
U_LTime U_MsgRd U_MsgWr U_Name U_PwdHist
U_PwdLc U_PwdTc U_Stat U_TimeOn

U_STAT(option:integer) :DATE or :INTEGER

Returns a statistic about the user that is tracked by PCBoard 5□□□
Valid values for "option" are 1 through 15

1 - Returns the first date the user called the system
2 - Returns the number of SysOp pages the user has requested
3 - Returns the number of group chats the user has participated in
4 - Returns the number of comments the user has left
5 - Returns the number of 300 bps connects
6 - Returns the number of 1200 bps connects
7 - Returns the bumber of 2400 bps connects
8 - Returns the number of 9600 bps connects
9 - Returns the number of 14400 bps connects
10 - Returns the number of security violations
11 - Returns the number of "not registered in conference" warnings

12 - Returns the number of times the users download limit has been reached
13 - Returns the number of "file not found" warnings
14 - Returns the number of password errors the user has had
15 - Returns the number of verify errors the user has had

See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful U_InConf U_LDate U_LDir U_Lmr U_Logons U_LTime U_MsgRd U_MsgWr U_Name U_PwdHist U_PwdLc U_PwdTc U_RecNum U_TimeOn

U_TIMEON() :INTEGER

Returns the current users time online today in minutes
See also : U_BdlU_BdlDay U_BulU_Fdl U_Ful U_InConf U_LDate U_LDir U_Lmr U_Logons U_LTime U_MsgRd U_MsgWr U_Name U_PwdHist U_PwdLc U_PwdTc U_RecNum U_Stat

VALCC(CCnum:string) :BOOLEAN

Returns TRUE if "CCnum" is a valid credit card number
See also : ValCC InputCC CcType

VALDATE(date:string) :BOOLEAN

Returns TRUE if "date" is in a valid date format
See also : ValTime

VALTIME(time:string) :BOOLEAN

Returns TRUE if "time" is in a valid time format
See also : ValDate

VER() :INTEGER

Returns the version number of PCBoard that is running

XOR(var1:integer, var2:integer) :INTEGER

Returns the bitwise exclusive-or of two integer expressions
See also : Or And Not

YEAR(var:date) :INTEGER

Returns the year (1900-2079) of "var"
See also : Month Day Dow

YESCHAR() :STRING

Returns the current language yes character
See also : NoChar

ADJTIME min:integer

Add or subtract "min" minutes to the users time available this session
See also : EvtTimeAdj

ANSIPOS col:integer, row:integer

If ANSI is available, position the cursor in column "col" and in row "row"
Legal ranges: 1 <= col <= 80
1 <= row <= 23 (Because of the status lines)
(1,1) is the top left corner of the screen

Note : Be aware that the user may have a different number of lines on his screen... if the user has 50 lines for exemple and that you do an ANSIPOS sentence to position the cursor on the 23rd line, the user will have a prompt in the middle of his screen...

See also : GetX GetY

BACKUP var:integer

Backup (move the cursor to the left) "var" columns without going past column 1
See also : Forward

BITCLEAR variable:multitype, bit:integer

Clears a specified bit from a variable.
This statement is primarily intended to be used with BIGSTR variables which can be up to 2048 bytes long. However, it will work with other data types as well if desired. Just be aware of the potential problems in 'bit twidling' non-string buffers and then trying to access them later as their 'intended' type without re-initializing the variable. If the bit parameter (an integer from 0 to the number of bits in the object) is invalid no processing takes place. □□□□
See also : BitSet IsBitSet

BITSET variable:multitype, bit:integer

Set a specified bit from a variable.
This statement is primarily intended to be used with BIGSTR variables which can be up to 2048 bytes long. However, it will work with other data types as well if desired. Just be aware of the potential problems in 'bit twidling' non-string buffers and then trying to access them later as their 'intended' type without re-initializing the variable. If the bit parameter (an integer from 0 to the number of bits in the object) is invalid no processing takes place. □□□
See also : BitClear IsBitSet

BLT bltnr:integer

Display bulletin number "bltnr"

BROADCAST var1:integer, var2:integer, message:string

Broadcast message "message" to nodes from "var1" to "var2" inclusive

□□□

BYE

Same as having the user type BYE from the command prompt
See also : Goodbye Hangup DtrOff

CALL ppename

Load and execute PPE filename specified by "ppename"
See also : Shell

CDCHKOFF

Turn off carrier detect checking
See also : CdCheckOn CdOn

CDCHKON

Turn on carrier detect checking
See also : CdCheckOff CdOn

CHAT

Initiate SysOp chat mode
See also : ChatStat PageStat

CLOSECAP

Close the capture file previously opened with OpenCap
See also OpenCap

CLREOL

Clear to the end of the line, with the current color if in ANSI mode
See also : Cls

□□□

CLS

Clear the screen, with the current color if in ANSI mode
See also : ClrEol

COLOR clr:integer

Change the current color to "clr"
See also : CurColor DefColor

CONFFLAG conf:integer, flags:integer

Turn on the conference "conf" flags specified by "flags"
See also : ConfSel ConfSys ConfMw CurConf ConfExp CurConf
ConfUnFlag Join ConfAlias LastIn ConfReg

CONFUNFLAG conf:integer, flags:integer

Turn off the conference "conf" flags specified by "flags"
See also : ConfSel ConfSys ConfMw CurConf ConfExp CurConf
ConfFlag Join ConfAlias LastIn ConfReg

DEC var:multitype

Decrement the value of var
See also : Inc

DELAY dlay:integer

Pause for "dlay" clock ticks (1 clock tick = 1/18.2 second)
See also : Wait

DELETE file:string

Deletes the filename specified by "file" (ERASE is a synonym)
See also : Copy Append Exist FileInf Rename

DELUSER

Flags the current user record for deletion

DIR arg:string

Performs a file directory command, passing it "arg" as arguments

DISPFILE file:string, flag:integer

Display file "file" with "flag" alternate file flags
valid flags : GRAPH
SEC
LANG
See also : DispStr

DISPSTR var:string

Display file if "var" is "%filename", execute PPE if "var" is "!filename", or display string "var"

DISPTEXT promptnr:integer, flagson:integer

Display PCBTEXT prompt "promptnr" using flags "flagson"
valid flags : NEWLINE
LFBEFORE
LFAFTER
BELL
LOGIT

LOGITLEFT
See also : DispFile

DOINTR intr, ax, bx, cx, dx, si, di, flags, ds, es (all is integer)

Generate interrupt number "intr" (0-255) with the register values passed as parameters

Note : Use DoIntr at your own risks !
See also : RegAx RegAh RegAl
RegBx RegBh RegBl
RegCx RegCh RegCl
RegDx RegDh RegDl
RegDi RegEs RegSi
RegDs RegCf RegF

DTROFF

Turn off the DTR signal
Note : on most modems, lowering DTR will cause modem to hangup... this is a good way if you want to simulate a bad connection, and then hangup without goodbye screens... This is the best way for you, the nice sysop, to free your line quickly... :)
See also : DtrOn Goodbye Bye Hangup

DTRON

Turn on the DTR signal
See also : DtrOff

□□□

END

End PPE execution
See also : End If End While

FAPPEND chnl:integer, file:string, access:integer, shrmod:integer

---

Use channel "chnl" to open file "file" in append mode with access mode "access" and share mode "shrmod"
valid channels: 0 - 7 [0 is used for script questionnaires]
valid access modes : O_RD, O_WR, O_RW [should use O_RW]
valid share modes : S_DN, S_DR, S_DW, S_DB
See also : FOpen FClose FCreate

FCLOSE chnl:integer

Close channel "chnl"
Accept channel -1 as the ReadLine() function 'channel' and close it
See also : FAppend FClose FCreate FFlush

FCREATE chnl:integer, file:string, access:integer, shrmod:integer

Use channel "chnl" to create and open file "file" in access mode "access" and share mode "shrmod"
   valid channels: 0 - 7 [0 is used for script questionnaires]
   valid access modes : O_RD, O_WR, O_RW [should use O_WR]
   valid share modes : S_DN, S_DR, S_DW, S_DB
See also : FOpen FClose FAppend

FGET chnl:integer, var:multitype

Read a line from channel "chnl" and assign it to "var"
See also : FPut FPutLn FPutPad FRead

FOPEN chnl:integer, file:string, access:integer, shrmod:integer

Use channel "chnl" to open file "file" in access mode "access" and share mode "shrmod"
   valid channels: 0 - 7 [0 is used for script questionnaires]
   valid access modes : O_RD, O_WR, O_RW
   valid share modes : S_DN, S_DR, S_DW, S_DB
See also : FCreate FClose FAppend FDefIn FDefOut

FOR ... NEXT

Usage :
FOR VAR = start:integer TO stop:integer [STEP incstep:integer]
...
NEXT

FOR - Initializes a loop by assigning "start" to VAR and continuing while VAR <= "stop" (if "incstep" >= 0) or VAR >= "stop" (if "incstep" < 0) (TO is required to separate "start" and "stop". If STEP (optional) is not specified "incstep" defaults to 1)
NEXT - Adds "incstep" to VAR, transfers control to the closest FOR statement, and marks the end of the FOR loop
See also : While...EndWhile If...Then

FORWARD var:integer

Move the cursor forward (to the right) "var" columns without going past column 80
See also : Backup

FPUT chnl:integer, str:string[, str:string...]

Write one or more "str" out to channel "chnl"
See also : FGet FPutLn FPutPad FWrite FWrite

FPUTLN chnl:integer[, str:string[, str:string...]]

Write zero or more "str" out to channel "chnl" and terminate with a carriage return/line feed pair
See also : FGet FPut FPutPad FRead FWrite

FPUTPAD chnl:integer, str:string, len:integer

Write out "str", padding or truncating to length "len" as needed, to channel "chnl"
See also : FGet FPut FPutLn FRead FWrite

FRESHLINE

If the cursor is not in column 1, do a newline
See also : NewLine NewLines

FREWIND chnl:integer

Rewind channel "chnl" after flushing buffers and committing the file to disk.
See also : FSeek

GETUSER

Fill the predefined variables (U_...) with current information from the user record

GOSUB LABEL

Transfer control to LABEL, marking the current PPE location for a future Return statement (GO SUB is a synonym)
See also : GoTo
□□
GOTO LABEL

Transfer control to LABEL (GO TO is a synonym)
See also : GoSub

GOODBYE

Same as having the user type G from the command prompt □□□
See also : Bye DtrOff Hangup

HANGUP

Hangup on the user without any notification
See also : Bye Goodbye DtrOff

IF ... THEN ... ELSE

Usage 1:
IF (exp:boolean) statement ...
Evaluate "exp" and, if true, execute statement; otherwise skip to the next statement
Usage 2:
IF (exp:boolean) THEN
...

```
ELSEIF (exp2:boolean) THEN
...
ELSE
...
ENDIF
```

IF - If expression cond is TRUE then this statement transfers control to the statement(s) following it, otherwise control is tranferred to the next ELSEIF, ELSE or ENDIF statement (requires THEN [or DO] after the condition)
ELSEIF - (optional) If expression cond is TRUE then this statement transfers control to the statements following it, otherwise control is tranferred to the next ELSEIF, ELSE or ENDIF statement There may be multiple ELSEIF statements between the IF and ELSE statements (ELSE IF is a synonym; nothing is required to come after the condition, although THEN [or DO] may appear for clarification and consistency in the source code)
ELSE - (optional) Separates the false portion of an IF/ELSEIF statement from the true portion
ENDIF - Ends an IF/ELSEIF/ELSE statement block (END IF is a synonym)
See also : While...EndWhile For...Next

INC var:multitype

Increment the value of "var"
See also : Dec

INPUT prompt:string, var:string

Display "prompt" and get input from user, assigning it to "var" (60 characters maximum)
See also : InputCC InputDate InputInt InputMoney InputStr
InputText InputTime InputYN

INPUTCC prompt:string, var:string, color:integer

Display "prompt" in color "color" and get a credit card formatted string from the user, assigning it to "var" (16 characters maximum, valid characters 0-9)
See also : InputInputDate InputInt InputMoney InputStr
InputText InputTime InputYN

INPUTDATE prompt:string, var:string, color:integer

Display "prompt" in color "color" and get a date formatted string from the user, assigning it to "var" (8 characters maximum, valid characters 0-9 - /)
See also : InputInputCC InputInt InputMoney InputStr
InputText InputTime InputYN

INPUTINT prompt:string, var:string, color:integer

Display "prompt" in color "color" and get an integer formatted string from the user, assigning it to "var" (11 characters maximum, valid characters 0-9)
See also : InputInputCC InputDate InputMoney InputStr
InputText InputTime InputYN

INPUTMONEY prompt:string, var:string, color:integer

Display "prompt" in color "color" and get a money formatted string from the user, assigning it to "var" (13 characters maximum, valid characters 0-9 $.)
See also : InputInputCC InputDate InputInt InputStr InputText InputTime InputYN

INPUTSTR...

Usage :

INPUTSTR prompt:string, var:string, color:integer, len:integer, valid:string, flags:string
Display "prompt" in color "color" and get a string (maximum length "len", valid characters "valid", flags "flags") from the user, assigning it to "var"
valid length : 1-256
valid characters : any string
valid flags : ECHODOTS
FIELDLEN
GUIDE
UPCASE
STACKED
ERASELINE
NEWLINE
LFBEFORE
LFAFTER
WORDWRAP
NOCLEAR
HIGHASCII
AUTO
YESNO
See also : InputInputCC InputDate InputInt InputMoney InputText InputTime InputYN

INPUTTEXT prompt:string, var:string, color:integer, len:integer

Display "prompt" in color "color" and get a string (maximum length "len") from the user, assigning it to "var"
See also : InputInputCC InputDate InputInt InputMoney InputStr InputTime InputYN

INPUTTIME prompt:string, var:string, color:integer

Display "prompt" in color "color" and get a time formatted string from the user, assigning it to "var" (8 characters maximum, valid characters 0-9 :)
See also : Input InputCC InputDate InputInt InputMoney InputStr InputText InputYN

INPUTYN prompt:string, var:string, color:integer

Display "prompt" in color "color" and get a yes/no response from the user, assigning it to "var" (1 characters maximum, valid characters determined by language)
See also : InputInputCC InputDate InputInt InputMoney InputStr InputText InputTime

JOIN conf:integer

Performs a join conference command, passing it "conf" as arguments
See also : ConfSel ConfSys ConfMwCurConf ConfExp CurConf ConfFlag ConfUnFlag ConfAlias LastIn ConfReg

KBDCHKOFF

Turn off keyboard time out checking

KBDCHKON

Turn on keyboard time out checking

KBDFILE file:string

Stuff the keyboard buffer with the contents of file "file"
See also : KbdBufSize PPLBufSize KbdFlush KbdStuff KbdString KbdFileUsed MdmFlush KeyFlush KbdFlush

KBDSTUFF str:string

Stuff the keyboard buffer with the contents of "str"
See also : KbdBufSize PPLBufSize KbdFlush KbdFile KbdString KbdFileUsed MdmFlush KeyFlush KbdFlush

LET var:multitype = EXP

Evaluate expression "EXP", convert and assign to "VAR"
NOTE: LET is the only optional keyword. If no keyword is found, LET is assumed. There are certain circumstances where it may be required, such as assignment to a variable named the same as a statement. PRINT, for example, would require a line such as LET PRINT = TRUE instead of just PRINT = TRUE)

LOG str:string, just:boolean

Write string "str" to the callers log, left justified if "just" is TRUE

MESSAGE...

Usage :
MESSAGE conf:integer, to:string, from:string, subject:string, sec:string, msgdate:date, retreceipt:boolean, echo:boolean, file:string

Write a message in conference "conf", to user "to" (empty string defaults to current caller), from user "from" (empty string defaults to current caller), subject "subject", security in "sec" (N or R; N is the default), pack out date in "msgdate" (0 for no pack out date), "retreceipt" True if return receipt desired, "echo" TRUE if message should be echoed, and "file" is the filename to use for the message text

MORE

Displays a MORE? prompt
See also : Wait Delay

MOUSEREG num,x1,y1,x2,y2,fontX,fontY,invert,clear,text

Set up a RIP mouse region on the remote terminal.
num = Is the RIP region number
x1,y1 = The (X,Y) coordinates of the upper-left of the region
x2,y2 = The (X,Y) coordinates of the lower-right of the region
fontX = The width of each character in pixels
fontY = The height of each character in pixels
invert = A boolean flag (TRUE to invert the region when clicked)
clear = A boolean flag (TRUE to clear and full screen the text window)
text = Text that the remote terminal should transmit when the region is clicked
See also : GrafMode

MPRINT str:string[, str:string...]

Display one or more string expressions on the callers screen only (this statement does not send anything to the BBS screen)
See also : MPrintLn Print PrintLn SPrint SPrintLn

MPRINTLN [str:string[, str:string...]]

Display zero or more string expressions on the callers screen only and follow with a newline (this statement does not send anything to the BBS screen)
See also : MPrint Print PrintLn SPrint SPrintLn

NEWLINE

Write a newline to the display □□□
See also : NewLines FreshLine

NEWLINES var

Write "var" newlines to the display □□□
See also : NewLine FreshLine

OPENCAP captfile:string, error:boolean

Open "captfile" and capture all screen output to it.
If an error occurs creating or opening "captfile", "error" is set to TRUE, otherwise "error" is set to FALSE.
See also : CloseCap

OPTEXT str:string

Writes string "str" into the @OPTEXT@ macro

PAGEOFF

Turn off the SysOp paged indicator (flashing p on status line)
See also : PageOn

PAGEON

Turn on the SysOp paged indicator (flashing p on status line)
See also : PageOff

POKEB addr:integer, val:integer

Assign the value "val" (0-255) to memory address "addr" (POKE is a synonym)□□□
See also : PeekB PeekDW PeekW PokeW PokeDW

POKEDW addr:integer, val:integer

Assign the value "val" (-2147483648 - +2147483647) to memory address "addr"
See also : PeekB PeekDW PeekW PokeB PokeW

POKEW addr:integer, val:integer

Assign the value "val" (0-65535) to memory address "addr"
See also : PeekB PeekDW PeekW PokeB PokeDW

POP var[,var...]

Pop values (previously pushed onto the stack) into a list of variables
See also : Push

PRFOUND & PRFOUNDLN

These work just like Print and PrintLn but, if the last SearchFind statement resulted in a match, it will automatically highlight found words.
See also : SearchFind

PRINT str:string[, str:string...]

Display one or more string expressions
See also : MPrint MPrintLn PrintLn SPrint SPrintLn

PRINTLN [str:string[, str:string...]]

Display zero or more string expressions and follow with a newline
See also : MPrint MPrintLn Print SPrint SPrintLn

PROMPTSTR prompt:integer, var:string, len:integer, valid:string, flags:integer

Display PCBTEXT entry "prompt" and get a string (maximum length "len", valid characters "valid", flags "flags") from the user, assigning it to "var"
valid length: 1-256
valid characters : any string
valid flags : ECHODOTS
FIELDLEN
GUIDE
UPCASE
STACKED
ERASELINE
NEWLINE
LFBEFORE
LFAFTER
WORDWRAP
NOCLEAR
HIGHASCII
AUTO
YESNO
See also : DispText

PUSH var[,var...]

Push a list of evaluated expressions onto the stack
See also : Pop

PUTUSER

Write the information from the predefined variables (U_...) to the user record
This statement is only intended to update user information if a successful GetUser or GetAltUser was issued previously. This was done to ensure that information for the current user wasn't written to another user or vice versa.
See also : GetUser

QUEST nr:integer

Do script questionnaire "nr"

RDUNET node:integer

Read information from USERNET.XXX for node "node"
See also : RdUsys WrUnet WrUsys

RDUSYS

Reads a USERS.SYS file, if present, and updates the users record
See also : RdUnet WrUnet WrUsys

RENAME oldname:string, newname:string

Rename file "oldname" to "newname"

See also : Delete Copy Append Exist FileInf

RESETDISP

Reset the display after an user abort

RESTSCRN

Restore the screen from the buffer previously saved with SaveScrn
See also : SaveScrn

RETURN

Return to the statement after the last GoSub or, if no GoSub is waiting for a RETURN, End the PPE

SAVESCRN

Save the current screen in a buffer for later restoration with the RestScrn
See also : RestScrn

SCRFILE lineVar, filenameVar

Find a file name and line number that is currently on the screen.
lineVar= Should be set before calling to the line number to start searching on (1 is the top line); Will be set to the line number where the file name was found or 0 if no file name was found filenameVar = Will be set to the file name if one is found on screen
See also : ScrText

SENDMODEM str:string

Send the text in "str" out to the modem

SEARCHINIT criteria, caseSensitive

Initialize search parameters for a faster BOYER-MOORE search algorithm.
criteria = A string expression with the search criteria in the same format used by PCBoard (ie, "THIS & THAT | BOB") caseSensitive = A boolean flag (TRUE to force a case sensitive search, FALSE otherwise)
See also : SearchFind PRFound/PRFoundLn SearchStop

SEARCHFIND bufferExpr, foundVar

Execute a BOYER-MOORE search on a text buffer using criteria previously defined with a SearchInit statement.
bufferExpr = The buffer to search foundVar = Set to TRUE if bufferExpr contains the search criteria, FALSE otherwise
See also : SearchInit PRFound/PRFoundLn SearchStop

SEARCHSTOP

Clears out previously entered search criteria. It takes no parameters.

See also : SearchInit SearchFind PRFound/PRFoundLn

SHELL com:boolean, code:integer, prog:string, arg:string

Shell (via COMMAND.COM if "com" is TRUE) to program/command "prog" with arguments "arg", saving the return value in "var"
NOTE: If "com" is TRUE, the value assigned to "var" will be the return code of COMMAND.COM, not "prog")
See also : Call

SHOWOFF

Turns off display of information to the screen □□□□
See also : ShowStat ShowOn

SHOWON

Turns on display of information to the screen
See also : ShowStat ShowOff

SORT sortArry, pointerArray

Sort the contents of an array into a pointer array.
sortArray = The data to sort (Any type may be used for this array)
pointerArray = An integer array which will be used as an array of pointers into sortArray for accessing sortArray in sorted order (This array should be of type INTEGER)

Note that sortArray and pointerArray are restricted to one (1) dimensional arrays.
The following is an example of displaying an array in unsorted and sorted order:

```
STRING  s(999) ; Remember that arrays are 0-based, so these statements
INTEGER p(999) ; will allocate 1000 elements each
; Do something here to read data into s
SORT s,p
INTEGER i
FOR i = 0 TO 999 ; This loop will display in unsorted order
  PRINTLN s(i)
NEXT
FOR i = 0 TO 999 ; This loop will display in sorted order
  PRINTLN s(p(i))
NEXT
```

SOUND freq:integer

Turn on the BBS PC speaker at the frequency (1-65535) specified by "freq" (or turn it off if the frequency is 0)

SPRINT str:string[, str:string...]

Display one or more string expressions on the BBS screen only (this statement does not send anything to the modem)
See also : MPrintLn MPrint Print PrintLn SPrintLn

SPRINTLN [str:string[, str:string...]]

Display zero or more string expressions on the BBS screen only and follow with a newline (this statement does not send anything to the modem)
See also : MPrintLn MPrint Print PrintLn SPrint

STARTDISP mode:integer

Start display monitoring in mode "mode"
valid modes : NC
FNS
FCL

STOP

Abort PPE execution without appending answers (channel 0) to the answer file
See also : End

TOKENIZE str:string

Tokenize string "string" into individual items separated by semi-colons or spaces
See also : GetToken TokenStr TokCount

TPAGET keyWord, infoVar

Get static information from a named TPA in string format.
keyword = The keyword of the TPA to use
infoVar = The variable into which to store the information
See also : Psa TPAPut TPACGet TPACPut TPARead TPAWrite TPACRead TPACWrite

TPAPUT keyWord, infoExpr

Put static information to a named TPA in string format.
keyword = The keyword of the TPA to use
infoExpr = The expression to write to store the TPA
See also : Psa TPACGet TPACPut TPARead TPAWrite TPACRead TPACWrite TPAGet

TPACGET keyWord, infoVar, confNum

Get information from a named TPA for a specified conference in string format.
keyword = The keyword of the TPA to use
infoVar = The variable into which to store the information
confNum = The conference number for which to retrieve information
See also : Psa TPAPut TPACPut TPARead TPAWrite TPACRead TPACWrite TPAGet

TPACPUT keyWord, infoExpr, confNum

Put information to a named TPA for a specified conference in string format.
keyword = The keyword of the TPA to use
infoExpr = The expression to write to store the TPA

confNum = The conference number for which to retrieve information
 See also : Psa TPAPut TPACGet TPARead
 TPAWrite TPACRead TPACWrite TPAGet

 TPAREAD keyWord, infoVar

Get static information from a named TPA.
keyword = The keyword of the TPA to use
infoVar = The variable into which to store the information
 See also : Psa TPAPut TPACGet TPACPut
 TPAWrite TPACRead TPACWrite TPAGet

 TPAWRITE keyWord, infoExpr

Put static information to a named TPA.
keyword = The keyword of the TPA to use
infoExpr = The expression to write to store the TPA
 See also : Psa TPAPut TPACGet TPACPut TPARead
 TPACRead TPACWrite TPAGet

 TPACREAD keyWord, infoVar, confNum

Get information from a named TPA for a specified conference.
keyword = The keyword of the TPA to use
infoVar = The variable into which to store the information
confNum = The conference number for which to retrieve information
 See also : Psa TPAPut TPACGet TPACPut TPARead
 TPAWrite TPACWrite TPAGet

 TPACWRITE keyWord, infoExpr, confNum

Put information to a named TPA for a specified conference.
keyword = The keyword of the TPA to use
infoExpr = The expression to write to store the TPA
confNum = The conference number for which to retrieve information
 See also : Psa TPAPut TPACGet TPACPut TPARead
 TPAWrite TPACRead TPAGet

 VARADDR var1:multitype, var2:integer

Assign the address (segment and offset) of "var1" to "var2"
 See also : VarSeg VarOff MkAddr

 VAROFF var1:multitype, var2:integer

Assign the offset address of "var1" to "var2"
 See also : VarSeg VarAddr MkAddr

 VARSEG var1:multitype, var2:integer

Assign the segment address of "var1" to "var2"
 See also : VarOff VarOff MkAddr

WAIT

Displays a PRESS ENTER TO CONTINUE? prompt
See also : More Delay Wait For

WAITFOR prompt:string, var:boolean, time:integer

Wait up to "time" seconds for the string "prompt", assigned TRUE to "var" if the string is found in the time specified or FALSE if the string is not found (WAIT FOR is a synonym)
See also : Wait

WHILE...

Usage 1:
WHILE (exp:boolean) statement ...
While "exp" is true execute statement; when "exp" is false execute following statements

□□□

Usage 2:

WHILE (exp) DO
...
ENDWHILE

WHILE - While "exp" is true execute statement(s); when "exp" is false transfer control to the first statement following the ENDWHILE statement (requires DO [or THEN] after the expression)
ENDWHILE - Transfers control to the closest WHILE statement and marks the end of the WHILE loop (END WHILE is a synonym)
See also : If..Then For...Next

WRUNET...

Usage :
WRUNET node:integer, nodestat:string, nodeusername:string,newnodecity:string, newoptext:string,broacasttext:string

Write information to USERNET.XXX for node "node", where "nodestat" is the new node status, "nodeusername" is the new node user name, "newnodecity" is the new node city, "newoptext" is the new node operation text, and "broadcasttext" is broadcast text
See also : RdUnet RdUsys WrUsys

WRUSYS

Writes (creates) a USERS.SYS file which can be used by a SHELLed application
See also : RdUnet RdUsys WrUnet

BREAK

Can be used to break out of a WHILE or FOR loop without the use of a GOTO statement
See also : Continue Quit

QUIT

Can be used to break out of a WHILE or FOR loop without the use of a GOTO statement (alias for BREAK)
See also : Continue Quit

CONTINUE

Can be used to abort the current iteration of a WHILE or FOR loop and resume with the next iteration of the loop □□□□
See also : Quit Break Loop

LOOP

Can be used to abort the current iteration of a WHILE or FOR loop and resume with the next iteration of the loop (alias for CONTINUE)
See also : Quit Break Continue

FFLUSH chnl:integer

flush a specified channels changes to disk
See also : FClose

FSEEK chnl:integer, byte:integer, position:integer

Position to any random location within a file
bytes is the number of bytes to move (+/-) relative to position
position is the base location to start the seek from
SEEK_SET (0) for the beginning of the file
SEEK_CUR (1) for the current file pointer location
SEEK_END (2) for the end of the file
See also : FRewind

FREAD chnl:integer, var:multitype, size:integer

Read binary data from a file
var is the variable into which data should be read
size is the size of data to read into var (0 - 2048)
See also : FGet

FWRITE chnl:integer, exp:multitype, size:integer

Write binary data to a file
exp is the expression whose result should be written
size is the size of data to write to var
See also : FPut FPutPad FPutLn

FDEFIN chnl:integer

Specify a default input file channel (used to speed up file input)
See also : FOpen

FDEFOUT chnl:integer

Specify a default output file channel (used to speed up file output)
See also : FOpen

FDGET var:multitype

Default channel input statement: use the exact same arguments as FGet except a channel parameter (the channel specified by FDefIn is assumed)
See also : FDPut FDPutPad FDPutLn FDRead FDWrite

FDREAD var:multitype, size:integer

Default channel input statement: use the exact same arguments as FRead except a channel parameter (the channel specified by FDefIn is assumed)

See also : FDPut FDPutPad FDPutLn FDGet FDWrite

FDPUT str:string[, str:string...]

Default channel output statement: use the exact same arguments as FPut except a channel parameter (the channel specified by FDefOut is assumed)
See also : FDRead FDPutPad FDPutLn FDGet FDWrite

FDPUTLN str:string[, str:string...]

Default channel output statement: use the exact same arguments as FPutLn except a channel parameter (the channel specified by FDefOut is assumed)
See also : FDRead FDPutPad FDPut FDGet FDWrite

FDPUTPAD str:string, len:integer

Default channel output statement: use the exact same arguments as FPutPad except a channel parameter (the channel specified by FDefOut is assumed)
See also : FDRead FDPutLn FDPut FDGet FDWrite

FDWRITE exp:multitype, size:integer

Default channel output statement: use the exact same arguments as FWrite except a channel parameter (the channel specified by FDefOut is assumed)
See also : FDRead FDPutPad FDPut FDGet FDPutPad

REDIM

Dynamically redimension an array at run-time
To use it you must declare the array in advance with the number subscripts desired. This allows the compiler to perform it's standard error checking on subscripts. For example:

```
STRING s(1,1,1)
REDIM s,5,5,5
LET s(4,4,4) = "Hello, World!"
PRINTLN s(4,4,4)
```

If an attempt is made to redimension an array with a different number of dimensions, an error or warning (as appropriate) will be generated.
See also : Compilation Options

APPEND srcfile:string, destfile:string

Append the contents of one file to another file.
ie:

```
APPEND "SRCFILE","DSTFILE"
```

See also : Delete Copy Exist FileInf Rename

COPY srcfile:string, destfile:string

Copy the contents of one file to another file.
ie:
□□□

```
COPY "SRCFILE","DSTFILE"
```

See also : Delete Append Exist FileInf Rename

LASTIN conf:integer

Set the users last conference in value. It can be used during the logon process to force the user into a particular conference at start up (for example, from a logon script).
See also : ConfSel ConfSys ConfMw CurConf ConfExp CurConf ConfFlag ConfUnFlag Join ConfAlias ConfReg

FLAG filepath:string

Allow flagging files for download directly from a PPE.
ie:
FLAG "C:\PATH\FILENAME.ZIP" ; Or whatever file name desired
Note that FLAG does not attempt to honor restrictions in the FSEC and/or DLPATH.LST files. This allows you to flag up any file desired.
See also : FlagCnt Download

DOWNLOAD cmd:string

Downloading files from PPL.
ie:
□□□□
DOWNLOAD "CMD;CMD;CMD"

The string passed to DOWNLOAD is a list of commands in the same format as what a user would type after a D or DB command.
If a file name for download is specified here it must be downloadable according to the criteria established in the FSEC and DLPATH.LST files.
If it is necessary to download a file not normally available via the FSEC and/or DLPATH.LST files the FLAG statement may be used to force it into the list of files to download.
See also : Flag

FLAGCNT()

Return the number of files flagged for download.
See also : Flag

WRUSYSDOOR str:string

Write a USERS.SYS file with a TPA record for a DOOR application.
ie:
□□□□
WRUSYSDOOR "DOORNAME"
See also : WrUsys

KBDSTRING str:string

Stuff strings to the keyboard (just like KbdStuff except 'keystrokes' are echoed to the display)

See also : KbdBufSize PPLBufSize KbdFlush KbdStuff KbdFile
KbdFileUsed MdmFlush KeyFlush KbdFlush

KBDFLUSH

Flush the local keyboard buffer and any stuffed keystroke buffers. It takes no arguments.
See also : KbdBufSize PPLBufSize KbdFlush KbdStuff KbdFile
KbdFileUsed MdmFlush KbdFlush KbdString

MDMFLUSH

Flush the incoming modem buffer. It takes no arguments.
See also : KbdBufSize PPLBufSize KbdFlush KbdStuff KbdFile
KbdFileUsed KeyFlush KbdFlush KbdString

KEYFLUSH

Flush both the local buffers and the incoming modem buffer. It takes no arguments.
See also : KbdBufSize PPLBufSize KbdFlush KbdStuff KbdFile
KbdFileUsed MdmFlush KbdFlush KbdString

ALIAS yesno:boolean

Allow PPE control of whether or not the user is using an alias
See also : PSA(1) UserAlias

ALIAS() :BOOLEAN

Return the users current ALIAS setting (TRUE = alias use on, FALSE = alias use off)
See also : TPAGet

CONFALIAS()

Return TRUE if the current conference is configured to allow aliases v□□□
See also : ConfSel ConfSys ConfMw CurConf ConfExp CurConf
ConfFlag ConfUnFlag Join LastIn ConfReg

USERALIAS() :BOOLEAN

Return TRUE if the current user is allowed to use an alias
See also : Alias

LANG

Change the language in use by the current user.
ie:
LANG langNum
See also : LangExt

ADJBYTES bytes:integer

Adjust the users total and daily download
To subtract bytes use a negative number for bytes. To add bytes use a positive number.
See also : AdjDBytes AdjTBytes AdjTFiles

ADJDBYTES bytes:integer

Adjust the users daily download bytes.
To subtract bytes use a negative number for bytes. To add bytes use a positive number.
See also : AdjTBytes AdjTFiles AdjBytes

ADJTBYTES bytes:integer

Adjust the users total download bytes.
To subtract bytes use a negative number for bytes. To add bytes use a positive number.
See also : AdjDBytes AdjTFiles AdjBytes

ADJTFILES files:integer

Adjust the users total download files.
To subtract files use a negative number for files. To add files use a positive number.
See also : AdjDBytes AdjTBytes AdjBytes

PUTALTUSER

Put user information. It is merely an alias for PutUser and may be used anywhere that PUTUSER would be used.
See also : GetAltUser

GETALTUSER user:integer

Get the information for an alternate user.
It will fill the user variables with information from the specified user record as well as redirect user statements and functions.
ie:
GETALTUSER userRecordNumber
If an attempt is made to get a record number that doesn't exist, the user functions will revert to the current user and the user variables will be invalidated as though no GetUser/GetAltUser statement had been issued (though they will continue to maintain any value held). PutUser/PutAltUser should be issued to commit any variable changes to the user record. Additionally, there is at least one statement that will not affect alternate users: AdjTime. It is restricted to the current user online. Also, if the alternate user is online, changes to the record won't take hold until after the user has logged off. Also, if there is not enough memory available (primarily for the last message read pointers) this statement will fail.
See also : PutAltUser

CURUSER() :INTEGER

Determine what users information, if any, is available via the user variables. It takes no arguments and returns one of the following values:
NO_USER (-1) - User variables are currently undefined
CUR_USER (0) - User variables are for the current user
Other - The record number of an alternate user for whom user variables are defined

Compilation Directives

;$INCLUDE:
Source files can be included from other source files. This is accomplished with a compiler directive in a comment like this:
;$INCLUDE:FILESPEC.EXT

Note that the first character need not be the semi-colon. An apostrophe ['] or asterisk [*] may also be used where appropriate.
This allows you to include subroutines from a source code 'library'. This should help in starting reusable code fragments. When the file is included, it is compiled as though it were in the main source file. For example:

```
FOO.INC
-------
:subroutine
PRINTLN "Hello!"
RETURN

FOO.PPS
-------
PRINTLN "Running FOO.PPS"
GOSUB subroutine
END ' This line is important!
*$INCLUDE:FOO.INC
```

Note the use of END in FOO.PPS. It is important in this case to ensure that you don't accidentally run subroutine twice by just falling through to it.
□□□
;$USEFUNCS
Allow you to specifiy that you want to use user-defined functions and procedures. This makes the code more flexible by allowing you to put your main code (code between Begin and End) anywhere in your program (usefull if you have to incude some user-defined functions with an include directive at the beginning of your code)

@Xnn Color Codes

PCBoard defines some macros to change color if user has ANSI capabilities.
If user doesn't support ANSI, PCB just skip those codes. It is a good way to colorize your screens and prompt because you don't have to check the ANSI flag, PCB deal with it automatically.
Color codes are made of 4 bytes. the first byte is used to enter in macro mode. The second indicates that we want to pass a color code. Next byte is the background color and the last is the foreground color.

BackGround codes :

0 - Black
1 - Blue
2 - Green
3 - Cyan
4 - Red
5 - Magenta
6 - Brown
7 - LightGray
8 - Black
9 - Blinking foreground on Blue background
A - Blinking foreground on Green background
B - Blinking foreground on Cyan background
C - Blinking foreground on Red background
D - Blinking foreground on Magenta background
E - Blinking foreground on Brown background
F - Blinking foreground on LightGray background

Foreground codes :

0 - Black
1 - Blue
2 - Green
3 - Cyan
4 - Red
5 - Magenta
6 - Brown
7 - LightGray
8 - DarkGray
9 - LightBlue
A - LightGreen
B - LightCyan
C - LightRed
D - LightMagenta
E - Yellow
F - White

GO ...

See : Go Sub Go To

... TO

See : Go To For...To...Next

PROCEDURE

[DECLARE] PROCEDURE proc([TYPE var1 [VAR]],...)

The keyword PROCEDURE is used in conjonction with the keyword DECLARE in the declaration of a user-defined procedure...
The optionnal VAR keyword tells PPL to copy the contents of the local variable back into the original variable when the procedure is finished processing.

The compiler directive ";$USEFUNCS" may be used in order to allow your main code (code between BEGIN & END) to be located anywhere within your file...
Example :

```
;$USEFUNCS
DECLARE PROCEDURE proc1(INTEGER i, STRING str, VAR INTEGER j)
INTEGER int1,int2
STRING s1
BEGIN
int1 = 1
int2 = 2
s1 = "HELLO"
proc1(int1,s1,int2)
PRINTLN "int1 =",int1
PRINTLN "int2 =",int2
PRINTLN "s1 =",s1
END
PROCEDURE proc1(INTEGER i,STRING str, VAR INTEGER j)
  PRINTLN "I'm in proc1"
  LET i = 30
  LET j = 15
ENDPROC
```

See also : Function

FUNCTION

[DECLARE] FUNCTION func(TYPE var1, ...) TYPE

The keyword PROCEDURE is used in conjonction with the keyword FUNCTION in the declaration of a user-defined function...
The compiler directive ";$USEFUNCS" may be used in order to allow your main code (code between BEGIN & END) to be located anywhere within your file...
The big difference between functions and procedures is that functions return a value. To assign the return value inside a function, simple use the name of the function just like a variable. You do not need to declare this variable, it is done for you. When the function is finished executing the value in the return variable will be made available as the return value.
Note that function calls can take place anywhere inside of an expression as well as stand-alone statements. This can be useful in situations when the functions return value is not needed, but the functions side effects are.
Example :

```
;$USEFUNCS
DECLARE FUNCTION Xto_theY(INTEGER x, INTEGER y) INTEGER
DECLARE FUNCTION square(INTEGER x) INTEGER
FUNCTION Xto_theY(INTEGER x, INTEGER y) INTEGER

INTEGER i
Xto_theY = x
for i = 2 to y
  Xto_theY = Xto_theY * x
next i
```

□□□
```
 ENDFUNC
 FUNCTION square(INTEGER x) INTEGER
   square = x * x
 ENDFUNC

 BEGIN
 PRINTLN "4 to the 3rd power = ",Xto_theY(4,3)
 PRINTLN "4 squared = ",square(4)
 END
```
 See also : Procedure

 DECLARE
 ■■■■■■■

 See : Function Procedure

 SELECT CASE
 ■■■■■■■■■■■

```
 SELECT CASE var
   CASE const1 [, const2..const3 [, expr ] ]
   .
   .
   .
   DEFAULT (or CASE ELSE)
 END SELECT
```
The SELECT CASE construct allows you to organize multiple execution paths into a clean, easy to read format.
Each CASE contains one or more expressions delimited by commas. Each CASE expression is compared to the SELECT CASE expression logically. If it is TRUE the body of the CASE is executed. The CASE body can contain as many statements as needed, including function calls. Note that ranges include the boundry values. eg 11..35 includes 11 and 35.
The DEFAULT case will be executed when none of the other CASE expressions evaluate to TRUE. For BASIC programmers, the CASE ELSE is also valid instead of DEFAULT.

Example :
```
 INTEGER i
    LET i = 3
SELECT CASE (i)
 CASE 1
   PRINTLN "i = 1"
   proc1(i)
 CASE 2,6,10
   PRINTLN "i is 2,6 or 10"
   proc2(i)
 CASE 3
   PRINTLN "i is 3"
 CASE 11..35
   PRINTLN "i is between 11 and 35"
 CASE 50..60,64,78
```

```
  PRINTLN "I is between 50 and 60 or 64 or 78
 DEFAULT
   PRINTLN "i is not a valid value"
END SELECT
```

DBASE III FUNCTIONS & STATEMENTS

PPL provide a load of functions & statements to access DBase III files...

STATEMENTS

```
DCREATE    channel,name,exclusive,fieldInfo ; create DBF file
DOPEN channel,name,exclusive ; open DBF file
DCLOSEchannel ; close DBF file
DSETALIAS  channel,name ; set DBF alias
DPACK channel ; pack DBF file
DLOCK channel ; lock DBF file
DLOCKFchannel ; lock DBF file (same as DLOCK)
DLOCKRchannel,recno; lock a record
DLOCKGchannel,recnos,count   ; lock a group of records
DUNLOCK    channel ; unlock any current locks
DNCREATE   channel,name,expression; create NDX file
DNOPENchannel,name ; open NDX file
DNCLOSE    channel,name ; close NDX file
DNCLOSEALL channel ; close all NDX files
DNEW  channel ; start a new record
DADD  channel ; add the new record
DAPPEND    channel ; append a blank record
DTOP  channel ; go to top record
DGO   channel,recno; go to specific record
DBOTTOM    channel ; go to bottom record
DSKIP channel,number    ; skip +/- a number of records
DBLANKchannel ; blank the record
DDELETE    channel ; delete the record
DRECALL    channel ; recall the record
DTAG  channel,name ; select a tag
DSEEK channel,expression; string or double
DFBLANK    channel,name ; blank a named field
DGET  channel,name,var  ; get a value from a named field
DPUT  channel,name,expression; put a value to a named field
DFCOPYchannel,name,channel,name   ; copy a field to a field
DCLOSEALL; close all DBF files
```

FUNCTIONS

```
DRECCOUNT (channel)   (INTEGER)   ; return the number of records
DRECNO    (channel)   (INTEGER)   ; return the current record number
DBOF (channel)   (BOOLEAN)   ; return the begin of file status
DEOF (channel)   (BOOLEAN)   ; return the end of file status
DDELETED  (channel)   (BOOLEAN)   ; return the deleted flag
DCHANGED  (channel)   (BOOLEAN)   ; return the changed flag
DFIELDS   (channel)   (INTEGER)   ; return count of fields
```

```
 DNAME(channel,number) (STRING)    ; return name of numbered field
 DTYPE(channel,name)  (STRING)    ; return type of named field
 DLENGTH  (channel,name)  (INTEGER)  ; return length of named field
 DDECIMALS (channel,name)  (INTEGER)  ; return decimals of named field
 DSELECT  (alias)(INTEGER)  ; returns channel assiciated with alias
 DSEEK(channel,expression) (INTEGER)    ; returns error status ( 0|1 )
; or seek success (0 = Error
; 1 = success, 2 = following record
; 3 = end of file )
 DGETALIAS (channel)  (STRING)  ; return the current alias
 DCLOSEALL  (BOOLEAN) error status   ; close all DBF files
 DOPEN(channel,name,exclusive)(BOOLEAN) error   ; open DBF file
 DCLOSE   (channel)  (BOOLEAN) error status   ; close DBF file
 DSETALIAS (channel,name)  (BOOLEAN) error status   ; set DBF alias
 DPACK(channel)  (BOOLEAN) error status   ; pack DBF file
 DLOCK(channel)  (BOOLEAN) error status   ; lock DBF file
 DLOCKR   (channel,recno)  (BOOLEAN) error status   ; lock a record
 DUNLOCK  (channel)  (BOOLEAN) error status   ; unlock any current locks
 DNOPEN   (channel,name)  (BOOLEAN) error status   ; open NDX file
 DNCLOSE  (channel,name)  (BOOLEAN) error status   ; close NDX file
 DNCLOSEALL(channel)  (BOOLEAN) error status   ; close all NDX files
 DNEW (channel)  (BOOLEAN) error status   ; start a new record
 DADD (channel)  (BOOLEAN) error status   ; add the new record
 DAPPEND  (channel)  (BOOLEAN) error status   ; append a blank record
 DTOP (channel)  (BOOLEAN) error status   ; go to top record
 DGO  (channel,recno)  (BOOLEAN) error status   ; go to specific record
 DBOTTOM  (channel)  (BOOLEAN) error status   ; go to bottom record
 DSKIP(channel,number) (BOOLEAN) error status   ; skip +/- a number of records
 DBLANK   (channel)  (BOOLEAN) error status   ; blank the record
 DDELETE  (channel)  (BOOLEAN) error status   ; delete the record
 DRECALL  (channel)  (BOOLEAN) error status   ; recall the record
 DTAG (channel,name)  (BOOLEAN) error status   ; select a tag
 DFBLANK  (channel,name)  (BOOLEAN) error status   ; blank a named field
 DGET (channel,name)  (STRING)  ; get a value from a named field
 DPUT (channel,name,expression)(BOOLEAN) error  ; put a value to a named field
 DFCOPY   (channel,name,channel,name)(BOOLEAN) error; copy a field to a field
 DERR (channel)  (BOOLEAN)   ; return error flag for channel
 DERRMSG  (errcode)  (STRING)    ; returns last DBase error text.
```

CAUTION: DBase functions that return the error status actually return !ERROR. This is to provide a consistent way to express an error in an expression. For example:

```
 if ( DERR (...)) println "Error!"  ;DERR  returns 1 or TRUE on an error.
 if (!DSEEK(...)) println "Seek failed!" ;DSEEK returns 0 or FALSE on an error.
```

NOTE: Where file names are used, file extensions are optional. Any extension you provide will be ignored. DBF and IDX are the default.

```
 channel  :  Any value between 0 and 7
 name:  Char string
 exclusive:  Integer (TRUE || FALSE)
 fieldinfo:  Character string with the following fields
   1- Field name
```

2- Field Type
C = Character
N = Numeric
F = Floating Point
D = Date
L = Logical
M = Memo
3- Field Length
4- Decimal (number of digits to the right of the decimal)

EXAMPLE:
```
string finfo(3)
let finfo(0) = "First,C,20,0"
let finfo(1) = "Last,C,20,0"
let finfo(2) = "Phone,C,15,0"
```

NOTE: multiple fields require an array of strings. 1 string for each field.
expression : Character String with search criteria on a field.

EXAMPLE:
```
string expr
let expr = "First"
recno,recnos,number,count : integers
```

DRIVESPACE ()

Usage: DRIVESPACE(drivespec) Return Val: Amount of divespace left of drive drivespec.
Example:
```
integer left
left = DRIVESPACE("c:\")
println "There are ",tostring(left)," bytes on drive C."
```
drivespec must include at least a drive letter AND a colon. Backslash is optional. With directory specs it will work also. valid drivespecs are C: C:\ C:\PCB These will all return drivespace left on dirve
*NOTE On LANTASTIC this will return drivespace of the current physical drive even if it is mapped as a directory. □□□□
See also : FileInf Delete

SETLMR

SETLMR conf#,msg#
Set the last read pointers for the specified conference.
Example :
```
Integer conf,msg
if(newuser == TRUE) then   ; If new user
  while(conf < HICONFNUM() ) DO; set all LMR's to
    join conf  ; HI_MSG - 10
    SETLMR conf,HIMSGNUM()-10
    INC conf
  Endwhile
endif
```

If conf# is greater than the number of actual confrences conf# will default to the highest conference number If msg# is greater than the highest message number in that conference, it will default to the highest message number in that conference. This could be used to set a new users mesg pointers to recent messages so they aren't replying to 3 years old messages. A useful feature would be to get the high conference number.
See also : HiConfNum HiMsgNum ActMsgNum LoMsgNum

SETENV

SETENV env_var
Set an environment variable
Example:

```
string s
let s = "stan=Stan"
SETENV s
.
.
.
if (GETENV("stan") = "Stan") then
Println "Environment variable stan = Stan "
Endif
```

Used to set DOS environment variable. This can be used for PPE's to communicate with other PPE's. The environment variables set within PPL will NOT be available to DOORs. Environment variables set within PPL will be cleared the next time PCBoard recycles through DOS.
See also : GetEnv, Shell, Call

FCLOSEALL

Closes all file channels
Example:

```
fopen 1, "Autoexec.bat"
fopen 2, "Config.sys"
.
.
.
fcloseall
```

See also : FOpen FClose FCreate FAppend FRewind FNext

FNEXT()

Returns an availble file channel. -1 when none are available.
Example: 8□□□

```
println "The next available file channel is ",FNEXT()
```

FNEXT was created in order to better support code libraries made possible by functions and procedures. File channel numbers can now be determined at runtime.
CAUTION: Until you actually OPEN a file FNEXT will return the same value over and over.
chan1 = FNEXT() chan2 = FNEXT() WRONG! chan1 will equal chan2
another gotcha: FOPEN FNEXT(),blah blah

There is no way to determine what channel was used to open the file!
Here's an example of how it should be used:
chan1 = FNEXT() FOPEN chan1,...
chan2 = FNEXT() FOPEN chan2,...
 See also : FOpen FClose FCreate FAppend FRewind FCloseAll

HICONFNUM()

Returns the highest conference number available on the board
Example: H□□□
 integer i
 println "The highest conference available is ",HICONFNUM()
If a conference is installed it will be counted, even if it is not being used.
 See also : SetLmr HiMsgNum ActMsgNum LoMsgNum

OUTBYTES()

Returns the number of bytes waiting in the modems output buffer Not available in local mode.
Example:
 integer i
 println "Bytes waiting in the modem output buffer ",OUTBYTES()
 See also : InBytes MGetByte SendModem MPrint MPrintLn MdmFlush

INBYTES()

Returns number of bytes waiting in the modem input buffer Not available in local mode.
Example:
 integer i
 Println "Bytes in modem input buffer = ",INBYTES()
 See also : OutBytes MGetByte SendModem MPrint MPrintLn MdmFlush

PCBMAC()

Returns a BIGSTR containing the expanded text of a PCB MACRO
Example:
 integer i,j, res
 j = PCBMAC("@Timelimit@")
 i = PCBMAC("@Timeused@")
 res = j-i
 println "You have ",res, " Minutes left"

 PCB MACROS not supported:
 @automore@ @beep@ @clreol@ @cls@ @delay@ @more@ @pause@ @poff@ @pon@ @pos@
 @qoff@ @qon@ @wait@ @who@ @x@

CRC32()

UNSIGNEDTYPE = CRC32(CRC_FILE,"C:\AUTOEXEC,BAT")
CRC32(CRC_STR,"Stan is super cool")
Returns an UNSIGNED value of the CRC of a file or string.

Example:
```
 Println "CRC on the file AUTOEXEC.BAT is", CRC32(CRC_FILE,"C:\AUTOEXEC.BAT")
```
The constants CRC_FILE and CRC_STR are the same as TRUE and FALSE. They were added to make it easier to see if a file or string was being processed.

ACTMSGNUM()

Returns number of active messages in current conference
Example:
```
 integer i
 println "There are ",ACTMSGNUM()," messages in conference ",CURCONF()
```
See also : Join HiConfNum LoMsgNum HiMsgNum

STACKLEFT()

Returns the number of bytes left on the *system* stack.
Example:
```
 println "There are ",STACKLEFT()," bytes left on the stack"
 ;recursive call support
 function stan(integer i,string str)
   if(stackleft() > STK_LIMIT) stan(i,"Debra")
 endfunc
```
This function was added to support nested and recursive function calls. Since function calls take a lot of stack space. As of now only about 26 nested or recursive calls can eat up the stack. This lets the programmer know when he/she is running out of stack space as to avoid a runtime error. Both recursion and nested function calls should check this value if more than just a few calls are to be executed. "□□□
See also : StackErr StackAbort

STACKERR()

Returns a boolean value which indicates a stack error has occured if TRUE.
Example:
```
 if (STACKERR()) then
println "An error has occured "
end
 endif
```
Because of the limited stack space for recursive function calls this function was created. It allows the programmer to determine if a stack error has occured while executing a PPE. This is in addition to the error message when the error occurs. The only way this will be useful is if the PPL programmer has told PPL not to abort on stack errors. PPL will *not* allow system memory to be corrupted when stack space has been exausted. It will disallow any more function calls when there is no system stack space left. *Note nested/recursive procedure calls are limited by heap space, not stack space.
See also : StackLeft StackAbort

STACKABORT

STACKABORT TRUE | FALSE
Example:
```
 STACKABORT TRUE ;Default is TRUE
```

This allows the programmer to tell the runtime module to try its best to continue executing after a stack error has occurred. If it is passed FALSE, it will abort execution after a stack error. If it is passed TRUE the PPE will continue to run. «□□□ CAUTION! If you continue to execute after a stack error, program execution will be unpredictable. PPL will not allow system memory to be corrupted because of a stack error.
See also : StackLeft StackErr

DNEXT()

Returns an available dbase file channel. -1 when none are available.
Example:
```
println "The next available dbase file channel is ",DNEXT()
```

DNEXT was created in order to better support code libraries made possible by functions and procedures. File channel numbers can now be determined at runtime. CAUTION! Until you actually OPEN a file DNEXT will return the same value over and over.
chan1 = DNEXT() chan2 = DNEXT() WRONG! chan1 will equal chan2
another gotcha: FOPEN DNEXT(),...
There is no way to determine what channel was used to open the file!
Here's an example of how it should be used:
```
chan1 = DNEXT() FOPEN   chan1,...
chan2 = DNEXT() FOPEN   chan2,...
```
See also : DBase functions

TODDATE (DATE date)

Converts any PPL type to DDATE type.
Example:
```
DATE d1
DDATE d2
d2 = TODDATE(d1)
```
This is an explicit type conversion. Implicit type conversion is also valid as with all other PPL types.
See also : Date DDate data type MkDate

FREALTUSER

Since only one GETALTUSER can be active at one time, FREALTUSER can allow other processes which need to use GETALTUSER (such as the MESSAGE commend) to do so.
Example:
```
string name
GETALTUSER 20
name = U_NAME()
FREALTUSER
message 1,name,...
```
See also : GetAltUser GetUser PutAltUser PutUser

ACCOUNTING

Several functions and statements have been added to support PCBoard accounting features. Also, many system constants have been added to make using these funtions and statements easier for the PPL programmer.

CONSTANTS

There are three new functions which return accounting information. Each function will return a value based on a parameter passed to it. Several constants have been added to make accessing these values easier. The following list details these consts and what they are used for.

for use with PCBACCOUNT() only!

val	const	Associated value
0	NEWBALANCE	Credits Given to a new user account
1	CHRG_CALL	Credits charged for a call
2	CHRG_TIME	Credits charged for time used (in minutes)
3	CHRG_PEAKTIME	Credits charged for peak time used
4	CHRG_CHAT	Credits charged for chat session
5	CHRG_MSGREAD	Credits charged for reading a message
6	CHRG_MSGCAP	Credits charged for capturing a message
7	CHRG_MSGWRITE	Credits charged for writing a message
8	CHRG_MSGECHOED	Credits charged for writing an echoed message
9	CHRG_MSGPRIVATE	Credits charged for writing a private message
10	CHRG_DOWNFILE	Credits charged for downloading a file
11	CHRG_DOWNBYTES	Credits charged for downloading bytes
12	PAY_UPFILE	Credits given for uploading a file
13	PAY_UPBYTES	Credits given for uploading bytes
14	WARN_LEVEL	Credit threshold for low credit warning

The following are for use with PCBACCSTAT() only!

val	constant	Associated value
0	ACC_STAT	Returns status of the "Enable Accounting" switch in the PWRD file. 0=Accounting disabled (N), 1=Tracking (T), and 2=Enabled (Y).
1	ACC_TIME	The amount of ADDITIONAL units to charge per minute while in the current conference.
2	ACC_MSGR	The amount to charge in ADDITION for each message read in the current conference.
3	ACC_MSGW	The amount to charge in ADDITION for each message entered in the current conference.

The following are for use with ACCOUNT(), ACCOUNT and RECORDUSAGE only!

val	constant	description	example
0	START_BAL	Users starting balance.	
1	START_SESSION	Users starting balance for this session	
2	DEB_CALL	Debit for this call	
3	DEB_TIME	Debit for time on	

4 DEB_MSGREADDebit for reading message
5 DEB_MSGCAP Debit for capturing a message
6 DEB_MSGWRITE Debit for writing a message
7 DEB_MSGECHOED Debit for echoed message
8 DEB_MSGPRIVATE Debit for writing private message
9 DEB_DOWNFILE Debit for downloading a file
10 DEB_DOWNBYTES Debit for downloading bytes
11 DEB_CHAT Debit for chat
12 DEB_TPU Debit for TPU
13 DEB_SPECIALDebit special
14 CRED_UPFILECredit for uploading a file
15 CRED_UPBYTES Credit for uploading bytes
16 CRED_SPECIAL Credit special
17 SEC_DROP Security level to drop to at 0 credits

This group of constants can be used to access or modify user account information using the ACCOUNT() function, ACCOUNT statement and/or RECORDUSAGE. The ACCOUNT() function returns the current value and the ACCOUNT statement is used to modify a value. Record usage also modifies a value with more information stored in a usage file.
See also : Account RecordUsage

ACCOUNT(INTEGER field)

ACCOUNT INTEGER field, INTEGER value

1) the ACCOUNT() function
Returns amount of credits charged for services corresponding to the field parameter.
Example:
println "You have been charged ",ACCOUNT(DEB_CHAT)," for chat"

field is the field number to access (1-14) or using DEB_ constants
See the Accounting section for a list of constants witch can be used with the ACCOUNT() function.
The account function is used to retrieve account information from PCBoard. These are the constants which can be used with the ACCOUNT() function.
2) The ACCOUNT Statement
field is a value between 0-14. Using system constants is reccomended. value is the amount of credits to add or subtract to field the field
Example:
ACCOUNT DEB_CHAT,10
The ACCOUNT statement is used to modify accounting information for a user. This statement will simply modify a debit value whereas the RECORDUSAGE will do the same thing as well as record information in the accounting file.
The valid constants for this statement are the same as those used for the ACCOUNT() Function. See the Accounting section for a list of those consts
See also : Accounting RecordUsage

□
RECORDUSAGE ...

Usage : RECORDUSAGE INTEGER field,STRING desc1,STRING desc2,DWORD unitcost,INTEGER value
Example:
RECORDUSAGE DEB_CHAT,"Debit for chat", "Using PPE",10,10
field is the field number to access (using DEB_... consts) descr1 is the descripttion of the charge descr2 is a subdescription of the charge unitcost is the cost per unit value is the number of units
Recordusage will update debit values in PCBoard as well as record descriptions and other information in an accounting file.
Valid values for the field parameter are 2-16. The constants corresponding with these values (DEB_???) could and should be used here. (see the Accounting section for a list of consts)
See also : Accounting Account PCBAccount PcbAccStat

PCBACCOUNT(INTEGER field)

Returns what PCBoard will charge a user for a certain activity. These are values the SysOp assigns in PCBsetup when accounting is configures and enabled.
Example: C□□□
println "You will be charged ",PCBACCOUNT(CHRG_CHAT)," for chat"

Valid values for the field paramter are 0-14. Use of the corresponding constants is encouraged. (see the Accounting section)
See also : Accounting Account RecordUsage PcbAccStat

PCBACCSTAT(INTEGER field)

Returns value in status field
Example:
PRINTLN "Mutiplier for credits is ",PCBACCSTAT(ACC_STAT)
This function can and should be used in conjunction with the ACC_??? constants as the field parameter. Valid values for field are 0-3. (see the Accounting section)
See also : Accounting Account RecordUsage RecordUsage

MESSAGE HEADER FIELD ACCESS CONSTANTS

Field Value	hex	dec	Field Description
HDR_ACTIVE	0x0E	14	Message active flag field
HDR_BLOCKS	0x04	4	Number of 128 byte blocks in message
HDR_DATE	0x05	5	Date message was written
HDR_ECHO	0x0F	15	Echoed message flag
HDR_FROM	0x0B	11	Who the message is from
HDR_MSGNUM	0x02	2	Message number
HDR_MSGREF	0x03	3	Reference message
HDR_PWD	0x0D	13	Message password
HDR_REPLY	0x0A	10	Message reply flag
HDR_RPLYDATE	0x08	8	Reply message date
HDR_RPLYTIME	0x09	9	Reply message time
HDR_STATUS	0x01	1	Message status
HDR_SUBJ	0x0C	12	Message subject

HDR_TIME 0x06 6Message time
HDR_TO 0x07 7Who the message is to.

These constants are for use with SCANMSGHDR(conf_num,start_msg,field,text) in the FIELD parameter.
See also : ScanMsgHdr

SCANMSGHDR(conf,start_msg,field,test)

Returns the first message number in the message base which matches the search criteria.
Example:
```
integer msgno
msgno = SCANMSGHDR(0,1,HDR_TO,"Stan")
```
This function can be used to scan PCBoard message bases for certain information. All fields in the standard header can be searched. There are 15 fields in the standard header. Valid values for field are 1-15. See the list of constants related to this function.
See also : MsgToFile Message Header Constats

MSGTOFILE conf,msg_no,filename

Writes a message into a file.
Example:
```
;Using SCANMSGHDR to search for a message
MSGTOFILE 0,200,"d:\msg1.txt"
DISPFILE "D:\msg1.txt",DEFS
```
This statement will take the given message and write it to a text file. The file's first 15 lines will contain standard header information. (One field per line) The headers are formatted to make parsing easier. The 16th line will state how many extended headers are present. The following line(s) will contain extended headers. (one per line) Finally, after the extended headers, will be a line containing "Message body:". Everything after that is the body of the message. 3□□□
See also : ScanMsgHdr, DispFile, HDR_... Consts

QWKLIMITS field,limit

QWKLIMITS(field)

1) The QWKLIMITS Statement
This statement allows the PPL programmer to modify a users QWK limits. Four fields can be modified with their statement.
Important note. You *must* use GET USER AND PUTUSER with these QWK functions.

Example:
```
GETUSER
QWKLIMITS MAXMSGS,500
PUTUSER
```

- Max Messages: Maximum messages allowed in a qwk packet
Note: If you specify a number higher than that contained in PCBSETUP the values in PCBSETUP will be used.

- Max Messages per Conference: Maximum messages allowed in a qwk packet per conference.
Note: If you specify a number higher than that contained in PCBSETUP the values in PCBSETUP will be used.
- Personal Attach Limit: Maximum number of bytes in attached files for the user.
- Public Attach Limit: Maximum number of bytes in attached files for the user.

Four constants have been defined to identify the FIELD value.

Constant	Value	Field
MAXMSGS	0	Max messages per qwk packet
CMAXMSGS	1	Max Messages per conference
ATTACH_LIM_U	2	Personal attach bytes limit
ATTACH_LIM_P	3	Public attach bytes limit

2) The QWKLIMITS() function
This functions returns the values contained in the users QWK configuration. The same constants used in the QWKLIMITS statements can be used with the field parameter.
Example:
```
GETUSER
PRINTLN QWKLIMITS(MAXMSGS)
```

MESSAGE HEADER FIELD ACCESS CONSTANTS

Field Value	hex	dec	Field Description
HDR_ACTIVE	0x0E	14	Message active flag field
HDR_BLOCKS	0x04	4	Number of 128 byte blocks in message
HDR_DATE	0x05	5	Date message was written
HDR_ECHO	0x0F	15	Echoed message flag
HDR_FROM	0x0B	11	Who the message is from
HDR_MSGNUM	0x02	2	Message number

HDR_MSGREF 0x03   3Reference message
HDR_PWD0x0D   13Message password
HDR_REPLY   0x0A   10Message reply flag
HDR_RPLYDATE0x08   8Reply message date
HDR_RPLYTIME0x09   9Reply message time
HDR_STATUS 0x01   1Message status
HDR_SUBJ   0x0C   12Message subject
HDR_TIME   0x06   6Message time
HDR_TO 0x07   7Who the message is to.

These constants are for use with SCANMSGHDR(conf_num,start_msg,field,text) in the FIELD parameter.

SCANMSGHDR(conf,start_msg,field,test)

Returns the first message number in the message base which matches the search criteria.
□□□
Example:
integer msgno msgno = SCANMSGHDR(0,1,HDR_TO,"Stan")
This function can be used to scan PCBoard message bases for certain information. All fields in the standard header can be searched. There are 15 fields in the standard header. Valid values for field are 1-15. See the list of constants related to this function.
See also : MsgToFile

MSGTOFILE conf,msg_no,filename

Writes a message into a file.
Example:
;Using SCANMSGHDR to search for a message MSGTOFILE 0,200,"d:\msg1.txt"
DISPFILE "D:\msg1.txt",DEFS
This statement will take the given message and write it to a text file. The file's first 15 lines will contain standard header information. (One field per line) The headers are formatted to make parsing easier. The 16th line will state how many extended headers are present. The following line(s) will contain extended headers. (one per line) Finally, after the extended headers, will be a line containing "Message body:". Everything after that is the body of the message.
See also : ScanMsgHdr, DispFile, HDR_... Consts

QWKLIMITS field,limit

QWKLIMITS(field)

1) The QWKLIMITS Statement
This statement allows the PPL programmer to modify a users QWK limits. Four fields can be modified with their statement.
Important note. You *must* use GET USER AND PUTUSER with these QWK functions.

Example:
GETUSER QWKLIMITS MAXMSGS,500 PUTUSER

- Max Messages: Maximum messages allowed in a qwk packet

* Note: If you specify a number higher than that contained in PCBSETUP the values in PCBSETUP will be used.

- Max Messages per Conference: Maximum messages allowed in a qwk packet per conference.
* Note: If you specify a number higher than that contained in PCBSETUP the values in PCBSETUP will be used.

- Personal Attach Limit: Maximum number of bytes in attached files for the user.
- Public Attach Limit: Maximum number of bytes in attached files for the user.

Four constants have been defined to identify the FIELD value.

Constant	Value	Field
MAXMSGS	0	Max messages per qwk packet
CMAXMSGS	1	Max Messages per conference
ATTACH_LIM_U	2	Personal attach bytes limit
ATTACH_LIM_P	3	Public attach bytes limit

2) The QWKLIMITS() function
This functions returns the values contained in the users QWK configuration. The same constants used in the QWKLIMITS statements can be used with the field parameter.
Example: GETUSER PRINTLN QWKLIMITS(MAXMSGS)