

OTHER PROCEDURES	
PACK(a,i,z)	UNPACK(z,a,i)
NEW(p)	DISPOSE(p)
ESCAPE	

EXTERNAL LIBRARY ROUTINES  
(portable)

```

PROCEDURE CALL$(ADDRESS:INTEGER; VAR A,F:
  BYTE; VAR BC, DE, HL, IX, IY:INTEGER);
FUNCTION FILE$STATUS(VAR F:TEXT):BYTE;
FUNCTION GETKEY:CHAR;
PROCEDURE HP$ERROR(NEWSTATE:BOOLEAN;
  VAR OLDSTATE:BOOLEAN);
PROCEDURE INKEY(VAR CH:CHAR;
  VAR READY:BOOLEAN);
FUNCTION INP(PORT:BYTE):BYTE;
PROCEDURE IO$ERROR(NEWSTATE:BOOLEAN;
  VAR OLDSTATE:BOOLEAN);
PROCEDURE OUT(PORT,VALUE:BYTE);
FUNCTION PEEK(ADDRESS:INTEGER):BYTE;
PROCEDURE POKE(ADDRESS:INTEGER; VALUE:BYTE);
PROCEDURE SETACNM(VAR F:TEXT; NAME:STRING);
PROCEDURE USER(ADDRESS:INTEGER;
  VAR HL:INTEGER);
PROCEDURE WRITECH(CH:CHAR);
PROCEDURE WRITESTRING(VAR S:CHARSTRING;
  FIRST, LAST:INTEGER);
PROCEDURE $MEMORY(VAR STACK, HEAP:INTEGER);

```

#### String Routines

```

FUNCTION CHARACTER(S:STRING; POSITION:
  INTEGER):CHAR;
FUNCTION CMPSTR(S1,S2:STRING): COMPAREVALUE;
FUNCTION CONC(S1,S2:STRING):STRING;
FUNCTION CPYSTR(S:STRING):STRING;
FUNCTION DECODED(S:STRING):REAL;
FUNCTION DECODEI(S:STRING):INTEGER;
FUNCTION DECODER(S:STRING):REAL;
FUNCTION DELETE(S:STRING; POSITION,
  LENGTH:INTEGER):STRING;
FUNCTION ENCODED(R:REAL):STRING;
FUNCTION ENCODEI(N:INTEGER):STRING;
FUNCTION ENCODER(R:REAL):STRING;
FUNCTION FIND(SUBSTRING,S:STRING):INTEGER;
FUNCTION INSERT(SUBSTRING,S:STRING;
  POSITION:INTEGER):STRING;
FUNCTION LEFT$(S:STRING; POSITION:
  INTEGER):STRING;
FUNCTION LEN(S:STRING):INTEGER;
FUNCTION MID$(S:STRING; POSITION,
  LENGTH:INTEGER):STRING;
FUNCTION REPLACE(OLDSTRING,NEWSTRING,S
  :STRING):STRING;
FUNCTION RIGHT$(S:STRING; POSITION:
  INTEGER):STRING;
FUNCTION STR$(LENGTH:INTEGER; CH:CHAR)
  :STRING;

```

	COMPILER ERROR MESSAGES
2	IDENTIFIER EXPECTED
3	'PROGRAM' EXPECTED
4	)' EXPECTED
5	:' EXPECTED
6	ILLEGAL SYMBOL
8	'OF' EXPECTED
9	(' EXPECTED
10	ERROR IN TYPE
11	LEFT BRACKET '[' OR '(' EXPECTED
12	RIGHT BRACKET ']' OR ')' EXPECTED
13	'END' EXPECTED
14	;' EXPECTED
15	INTEGER EXPECTED
16	'=' EXPECTED
17	'BEGIN' EXPECTED
20	',' EXPECTED
22	..' EXPECTED
23	.' EXPECTED
49	'ARRAY' EXPECTED
50	CONSTANT EXPECTED'
51	':=' EXPECTED
52	'THEN' EXPECTED
53	'UNTIL' EXPECTED
54	'DO' EXPECTED
55	'TO'/'DOWNT0' EXPECTED
57	'FILE' EXPECTED
58	INVALID OR MISSING OPERAND
62	DECIMAL PLACES ALLOWED ONLY FOR REAL
66	TYPE IDENTIFIER EXPECTED
80	OPEN COMMENT WITHIN A COMMENT
81	UNKNOWN OPTION
82	# REQUIRES A 2 CHARACTER HEX VALUE OR ##
101	IDENTIFIER DECLARED TWICE
102	LOWER BOUND EXCEEDS UPPER BOUND
103	IDENTIFIER IS NOT OF APPROPRIATE CLASS
104	UNDECLARED IDENTIFIER
105	CLASS OF IDENTIFIER IS NOT VARIABLE
107	INCOMPATIBLE SUBRANGE TYPES
113	ARRAY BOUNDS MUST BE SCALAR
117	UNSATISFIED FORWARD REFERENCE TO A TYPE IDENTIFIER OF A POINTER
119	;' EXPECTED (PARAMETER LIST NOT ALLOWED)
120	FUNCTION RESULT MUST BE SCALAR, SUBRANGE, OR POINTER
123	FUNCTION RESULT EXPECTED
126	IMPROPER NUMBER OF PARAMETERS
127	TYPE OF ACTUAL PARAMETER DOES NOT MATCH FORMAL PARAMETER
129	TYPE CONFLICT OF OPERANDS IN AN EXPRESSION
132	COMPARISON WITH '>' OR '<' NOT ALLOWED ON SETS
134	ILLEGAL TYPE OF OPERANDS
135	TYPE OF EXPRESSION MUST BE BOOLEAN
136	SET ELEMENT TYPE MUST BE SOME ENUMERATION TYPE

# COMPILER ERROR MESSAGES

```

138 TYPE OF VARIABLE IS NOT ARRAY
140 TYPE OF VARIABLE IS NOT RECORD
141 TYPE OF VARIABLE IS NOT POINTER
148 SET BOUNDS OUT OF RANGE
152 NO SUCH FIELD IN THIS RECORD
154 ACTUAL PARAMETER MUST BE A VARIABLE
156 MULTIDEFINED CASE LABEL
161 PROCEDURE OR FUNCTION ALREADY DECLARED
    AT A PREVIOUS LEVEL
165 LABEL ALREADY DEFINED
167 UNDECLARED LABEL
168 LABEL NOT DEFINED
182 "FOR" EXPRESSION MUST BE OF SOME
    ENUMERATION TYPE
183 "CASE" EXPRESSION MUST BE OF SOME
    ENUMERATION TYPE
184 "FOR" VARIABLE MUST BE LOCAL
185 OPERATION DEFINED FOR TEXT ONLY
186 OPERATION NOT DEFINED FOR TEXT FILES
193 ACCESS STATEMENT MISSING FOR COMMON
199 FEATURE NOT IMPLEMENTED
202 STRING CONSTANT CANNOT SPAN LINES
203 INTEGER CONSTANT TOO LARGE
210 FIELD WIDTH MUST BE INTEGER
211 FRACTION LENGTH MUST BE OF TYPE INTEGER
212 HEX FORMAT ALLOWED ONLY FOR TYPE INTEGER
219 PARAMETER MUST BE OF TYPE FILE
220 PARAMETER MUST BE OF TYPE INTEGER
223 PARAMETER MUST BE OF TYPE POINTER
230 ILLEGAL TYPE OF PARAMETER IN STANDARD
    PROCEDURE CALL
250 TOO MANY NESTED SCOPES - LIMIT IS 15
    
```

# RUNTIME ERROR MESSAGES

```

01 Out of stack space
02 Heap space exhausted
03 Invalid pointer in dispose
04 Illegal static nesting level
05 Attempt to divide by zero
06 Pcode instruction not defined
07 Sets not compatible
08 Undefined internal procedure
09 Physical IO error
DA Set element too large
10 Invalid array index
11 Attempt to read invalid number
12 Undefined file buffer variable
13 Arithmetic overflow
EB Attempt to write to input file
EC File not open
ED Attempt to read from output file
EE No heap space for file buffer
    
```

# EDIT <filename> KEY MAPPED COMMANDS

Key	Mnemonic	Function
^E	UP	Cursor up
^X	DN	Cursor down
^S	LF	Cursor left
^D	RT	Cursor right
^QS	BL	Cursor to BOLN
^QD	EL	Cursor to EOLN
^H	LF	Cursor left
^QH	HM	Cursor home
^F	FW	Forward by word
^A	BW	Backward by word
^I	TB	Tab forward
^QI	BT	Tab backward
^R	RU	Roll up
^C	RD	Roll down
^QR	TP	Top of buffer
^QC	BB	Bottom of buffer
^G	DC	Delete character
^T	DW	Delete word
^K	DE	Delete to end of line
^Y	DL	Delete line
^QY	UL	Undelete line
^QF	CL	Center line
^U	DU	Duplicate line
^O	SP	Split line
^P	MG	Merge line
^N	IL	Insert line
^M	NL	Next line
^J	FN	Find next
^QJ	FS	Find string
^L	RN	Replace next
^QL	RS	Replace string
^BM	MK	Mark block
^BC	CB	Copy marked block
^BD	DB	Delete marked block
^BI	IB	Insert block
^BF	FI	Fill marked block
^BJ	JF	Justify marked block
^BU	UR	Upper case marked block
^BL	LR	Lower case marked block
^BG	GM	Go to mark
^BS	SW	Swap cursor and mark
^BP	PR	Print marked block
^V	IC	Insert character mode
^Z	CM	Command mode

# COMMAND MODE

HELP	HP	Display help files
APPEND	AP	Append lines to buffer
WRITE	WR	Write lines to file
INSFILE	IF	Insert file
EXTRACT	XT	Write marked block to file
SAVE	SV	Save current changes
EXIT	EX	Exit and save changes
QUIT	QT	Exit without saving

<p>PASCAL PROGRAM STRUCTURE</p> <p>A parameter list may be defined as follows:</p> <pre>( &lt;VAR&gt; {identifier list} : {type};   &lt;VAR&gt; {identifier list} : {type} )</pre> <p>An identifier must start with a letter, and is composed of the letters, digits and the '_', '\$' characters. The first 8 characters must form a unique name.</p>	
--	--

<p>PREDEFINED TYPES</p> <p>CHAR, INTEGER, BOOLEAN, REAL, TEXT</p> <p>A type definition is:</p> <pre>(1) {predefined type}; (2) {constant}..{Constant}; (3) ( {identifier list} ); (4) &lt;packed&gt; RECORD       {identifier list} : {type};       {identifier list} : {type};       END; (5) &lt;packed&gt; ARRAY [ {bounds} ]       OF {type}; (6) FILE OF {type}; (7) SET OF {type};</pre>	
--	--

<p>FLOW CONTROL STATEMENTS</p> <pre>CASE {expression} OF   {selector list} : {statement};   {selector list} : {statement}; &lt; OTHERWISE {statement}; &gt; END;</pre> <pre>IF (expression) THEN   {statement} &lt; ELSE {statement}; &gt;</pre> <pre>REPEAT {statement list}; UNTIL { (expression) };</pre> <pre>WHILE { (expression) } DO {statement};</pre> <pre>GOTO {label};</pre> <pre>ESCAPE;</pre>	
--	--

<p>BOOLEAN OPERATORS</p>	
Operator	Result of Operation
OR	True if either one or both of the operands is true.
AND	True only if both operands are true.
NOT	True if operand is false.

<p>ARITHMETIC OPERATORS</p> <p>I = integer , R = real</p>			
Operator	Operation	Type Operands	Type Result
+	add	I , R	I , R
	set union	sets of compatible type	same type as larger set oper
-	subtract	I , R	I , R
	set diff	sets of compatible type	same type as larger set oper
*	multiply	I , R	I , R
	set intersection	sets of compatible type	same type as larger set oper
/	division	I , R	R
DIV	truncated division	I	I
MOD	modulus	I	I

# SPECIAL SYMBOLS

Symbols with only one representation:

+		-	*	/	
=	<>	<	<=	>=	>
(	)	'	:=	.	,
;	:	#	\	::	

Symbols with alternate representations:

symbol	alternate	Meaning
{	(*	open comment
}	*)	close comment
^	@	pointer symbol
[	(.	open array index
}	.)	close array index

<p>RELATIONAL OPERATORS</p>	
Operator	Result of Operation
=	True if left operand is equal to right.
<>	True if left operand is not equal to right.
<	True if left operand is less than right.
>	True if left operand is greater than right.
<=	True if left operand is less than or equal to right.
>=	True if left operand is greater than or equal to right.

<p>OPERATOR PRECEDENCE</p> <pre>()      &lt;-- highest precedence + , -    &lt;-- unary operators * , / , DIV , MOD + , - = , &lt;&gt; , &lt; , &gt; , &lt;= , &gt;= , IN NOT AND OR      &lt;-- lowest precedence</pre>	
--	--

RESERVED WORDS				
The following list of words are keywords and have special meaning in a program. They may not be used as identifiers.				
AND	DOWNT0	IF	OR	THEN
ARRAY	ELSE	IN	PACKED	TO
BEGIN	END	LABEL	PROCEDURE	TYPE
CASE	FILE	MOD	PROGRAM	UNTIL
CONST	FOR	NIL	RECORD	VAR
DIV	FUNCTION	NOT	REPEAT	WHILE
DO	GOTO	OF	SET	WITH

<p>ARITHMETIC FUNCTIONS</p>			
ABS(x)	absolute value	SQR(x)	square
SIN(x)	sine	COS(x)	cosine
ARCTAN(x)	arctangent	EXP(x)	natural
LN(x)	natural logarithm	EXPONENTIAL	exponential
		SQRT(x)	sq. root

<p>FILE ASSOCIATED PROCEDURES</p>	
RESET(f)	REWRITE(f)
PAGE(f)	CLOSE(f)
MESSAGE(s)	GET, PUT
READ , READLN	WRITE, WRITELN

<p>FUNCTIONS</p>		
Boolean	Transfer	
ODD(x)	TRUNC(x)	LOCATION(x)
EOLN(f)	ROUND(x)	SIZE(x)
EOF(f)	ORD(x)	HB(x)
	CHR(x)	LB(x)

<p>OTHER FUNCTIONS</p>	
SUCC(x)	pred(x)

(optional item , {user defined item}  
"items" are computer responses

#### COMPILING

Short Form:

PASCAL {FILE CONTAINING SOURCE}

Long Form:

PASCAL  
"SOURCE =" <FILE CONTAINING SOURCE>  
"LISTING=" <FILE FOR LISTING>  
"OBJECT =" <FILE FOR OBJECT>

#### EXECUTING

Load and Execute:

RUNP {FILENAME} <STACK SIZE>

Execute or Build Command File:

LINKLOAD

Command		Meaning
L	-	load object file
R	-	run program
T	-	return to TRSDOS
I	-	clear symbol table
S	-	display symbol table
B	-	build command file

#### DEVICE NAMES

Name	Device
:L	line printer
:C	crt
:D	dummy

#### ASCII CHARACTER SET

Dec- imal	Hex	Name	Dec- imal	Hex	Name
0	00	NUL	11	0B	VT
1	01	SOH	12	0C	FF
2	02	STX	13	0D	CR
3	03	ETX	14	0E	SO
4	04	EOT	15	0F	SI
5	05	ENQ	16	10	DLE
6	06	ACK	17	11	DC1
7	07	BEL	18	12	DC2
8	08	BS	19	13	DC3
9	09	HT	20	14	DC4
10	0A	LF	21	15	NAK

#### ASCII Character Set

Dec- imal	Hex	Name	Dec- imal	Hex	Name
22	16	SYN	75	4B	K
23	17	ETB	76	4C	L
24	18	CAN	77	4D	M
25	19	EM	78	4E	N
26	1A	SUB	79	4F	o
27	1B	ESC	80	50	P
28	1C	FS	81	51	Q
29	1D	GS	82	52	R
30	1E	RS	83	53	S
31	1F	US	84	54	T
32	20	" "	85	55	U
33	21	!	86	56	V
34	22	"	87	57	w
35	23	#	88	58	X
36	24	\$	89	59	O
37	25	%	90	5A	z
38	26	&	91	5B	f
39	27	'	92	5C	\
40	28	(	93	5D	]
41	29	)	94	5E	^
42	2A	*	95	5F	
43	2B	+	96	60	~
44	2C	,	97	61	a
45	2D	-	98	62	b
46	2E	.	99	63	c
47	2F	/	100	64	d
48	30	0	101	65	e
49	31	1	102	66	f
50	32	2	103	67	g
51	33	3	104	68	h
52	34	4	105	69	i
53	35	5	106	6A	j
54	36	6	107	6B	k
55	37	7	108	6C	l
56	38	8	109	6D	m
57	39	9	110	6E	n
58	3A	:	111	6F	o
59	3B	;	112	70	p
60	3C	<	113	71	q
61	3D	=	114	72	r
62	3E	>	115	73	s
63	3F	?	116	74	t
64	40	@	117	75	u
65	41	A	118	76	v
66	42	B	119	77	w
67	43	C	120	78	x
68	44	D	121	79	y
69	45	E	122	7A	z
70	46	F	123	7B	{
71	47	G	124	7C	
72	48	H	125	7D	}
73	49	I	126	7E	~
74	4A	J	127	7F	DEL

# TRS-80® PASCAL

## Quick Reference Guide

#### PASCAL PROGRAM STRUCTURE

<optional items> , {user defined}

Other items must appear as printed.  
Multiple LABEL, TYPE, CONST, COMMON,  
ACCESS, VAR, PROCEDURE and FUNCTION  
declarations are allowed.

```
PROGRAM {program name};  
<LABEL {label list};>  
<TYPE {identifier} = {type definition}>  
<CONST {identifier} = {value};>  
<COMMON {identifier} : {type};>  
<ACCESS {identifier list};>  
<VAR {identifier} : {type};>  
  
<PROCEDURE {identifier} {parameter list};  
<FORWARD;> <EXTERNAL;>  
<{If not declared FORWARD or EXTERNAL  
then declarations and body here} >; >  
  
<FUNCTION {identifier} {parameter list};  
{type returned result};  
<{If not declared FORWARD or EXTERNAL  
then declarations and body here} >; >  
  
BEGIN  
{program body}  
END.
```

#### MODEL 4

Manufactured for Radio Shack  
by Alcor Systems  
Copyright (C) 1983 by Alcor Systems