Appendix B − BASIC ROM summary

BASIC1	BASIC2	ROUTINE
8000	8000	BASIC entry point
8006	8006	Paged ROM data
801F	8023	Language initialisation
806D	8071	Keyword table
835A	836D	Keyword action address table
843C	8451	Assembler mnemonic tables
84E6	84FD	‘]’ (Back to BASIC from assembler)
84ED	8504	‘[’ statement (Assembler entry point)
87E4	8821	Evaluate integer <numeric>
87FD	887C	Substitute token in buffer
8819	8897	Tokenise line number
88AB	8926	Check for alphanumeric char (or ‘.’)
88D3	8951	Tokenise a line
8A13	8A8C	Get character at PTRB
8AlE	8A97	Get character at PTRA
8A3D	8AB6	‘OLD’ statement
8A50	8AC8	‘END’ statement
8A59	8AD0	‘STOP’ statement
8A7D	8ADA	‘NEW’ statement
8A80	8ADD	Cold start
8A96	8AF3	W arm start
8A99	8AF6	Enter immediate mode
8BAA	8B47	‘=’ statement (return FN value)
8BC3	8B73	‘*’ statement (send line to OSCLI)
8AED	8B7D	‘DATA’,’DEF’, ’REM’ statement (skip line)
8B0C	8B9B	Continue execution at next statement
8B57	8BE4	‘LET’ statement
8BD0	8C1E	Assign string
8C5B	8CC1	Pop parameter value
8CC5	8D2B	‘PRINT#’ statement
8D33	8D9A	‘PRINT’ statement
8DBD	8E2A	‘TAB(X,Y)’ in printable section
8DD9	8E40	‘TAB(’ in printable section
8DF2	8E58	‘SPC’ in printable section
8E57	8EBD	‘CLG’ statement
8E5E	8EC4	‘CLS’ statement
8E6C	8ED2	‘CALL’ statement
8ECE	8F31	‘DELETE’ statement

8F37	8FA3	‘RENUMBER’ statement
905F	90AC	‘AUTO’ statement
90DD	912F	‘DIM’ statement
91EB	9236	Perform ‘space required’ multiplication
9212	925D	‘HIMEM’ statement
9224	926F	‘LOMEM’ statement
9239	9283	‘PAGE’ statement
9326	928D	‘CLEAR’ statement
9243	929F	‘TRACE’ statement
927B	92C9	‘TIME’ statement
9292	92E3	Get integer <factor>
92AC	92EB	Get real <factor>
92B6	9304	‘PROC’ statement
92D5	9323	‘LOCAL’ statement
9310	9356	‘ENDPROC’ statement
932F	937A	‘GCOL’ statement
9346	938E	‘COLOUR’ statement
935A	939A	‘MODE’ statement
93A1	93E4	‘MOVE’ statement
93A5	93E8	‘DRAW’ statement
93AE	93F1	‘PLOT’ statement
93EF	942F	‘VDU’ statement
941B	945B	Look for FN/PROC in list
9429	9469	Look for variable in list
94AD	94ED	Link in new PROC/FN
94BC	94FC	Link in new variable
94F7	9531	Clear space for information block
951F	9559	Scan variable name
9548	9582	Find variable, creating if needed
9595	95C9	Find variable at PTRA
95A9	95DD	Find variable at PTRB
97AC	97DD	Get tokenised line number at PTRA
97D6	9807	Set PTRB to PTRA, then. ..
97E2	9813	Evaluate <numeric> after ‘=’
980B	9852	Check end of statement at PTRB
9810	9857	Check end of statement at PTRA
9851	9880	Move to start of next statement
9893	98C2	‘IF’ statement
98F1	991F	Print line number in IntA
9942	9970	Look for program line
99C0	99EA	Perform integer division
9A76	9A9E	Perform comparison
9AF7	9B1D	Set PTRB to PTRA, then...

9B03	9B29	Get <numeric> or <string> at PTRB
9B14	9B3A	'OR' operator
9B2F	9B55	'EOR' operator
9B45	9B72	Get <logical expression>
9B54	9B7A	'AND' operator
9B76	9B9C	Get <relnl expression>
9B88	9BAE	'=' operator (comparison)
9BA7	9BCD	'<' operator
9BAE	9BD4	'<=' operator
9BB9	9BDF	'<>' operator
9BCB	9BE8	'>' operator
9BD4	9BFA	'>=' operator
9C1D	9C42	Get <expression>
9C29	9C4E	'+' operator
9C90	9CB5	'−' operator
9D17	9D3C	'*' operator
9DAE	9DD1	Get <term>
9DC2	9DE5	'/' operator
9DDE	9E01	'MOD' operator
9DE7	9E0A	'DIV' operator
9DFD	9E20	Get <sub-term>
9E12	9E35	'↑' operator (exponentiation)
9E81	9E90	Convert number to HEX string
9ED0	9EDF	Convert number to string
A06C	A07B	Get number at PTRB
A169	A178	Add FPB mantissa to FPA mantissa
A188	A197	Multiply FPA mantissa by 10
A1CB	A1DA	Test FP A
A1E5	A1F4	Multiply FPA by 10
A20F	A21E	Copy FPA into FPB
A23E	A24D	Divide FPA by 10
A295	A2A4	Add A to PEA mantissa
A2AF	A2BE	Convert IntA to FPA
A2DE	A2ED	Convert A to FPA
A2F4	A303	Normalise FPA
A33F	A34E	Load FPB from packed number at (&4B)
A36E	A37D	Store FPA at &471 −&475
A372	A381	Store FPA at &476−&47 A
A376	A385	Store FPA at &46C−&470
A37E	A38D	Store FPA at (&4B)
A3A3	A3B2	Load FPA from &46C−&470
A3A6	A3B5	Load FPA from (&4B)
A3F2	A3E4	Convert FPA to IntA

A40C	A3FE	Convert FPA to fixed format
A463	A453	Set FPB to zero
A494	A486	Extract fractional part of FPA
A505	A4D0	Subtract number at (&4B) from FP A
A4DE	A4D6	Exchange FPA with number at (&4B)
A4E4	A4DC	Copy FPB into FPA
A50B	A4FD	Subtract FPA from number at (&4B)
A50E	A500	Add number at (&4B) to FPA
A513	A50B	Add FPB to FPA
A611	A606	Multiply FPA by number at (&4B)
A6lE	A613	Multiply FPA by FPB
A661	A656	Multiply FPA by (&4B); test for o'flow
A691	A686	Set FPA to zero
A6A4	A699	Set FPA to 1
A6B0	A6A5	Invert FPA (set FPA = 1/FPA)
A6B8	A6AD	Divide (&4B) by FPA
A6C9	A6BE	'TAN' function
A6F2	A6E7	Divide FPA by (&4B)
A6FC	A6F1	Divide FPA by FPB
A7B4	A7B4	'SQR' function
A7EF	A7E9	Point &4B ,&4C at &47B
A7F3	A7ED	Point &4B,&4C at &471
A7F7	A7F1	Point &4B,&4C at &476
A7FB	A7F5	Point &4B,&4C at &46C
A804	A7FE	'LN' function
A856	A869	Constant: log(e) (i.e. 'LOG EXP 1')
A85B	A86E	Constant: ln(2)
A860	A873	Constant series for 'LN' evaluation
A889	A897	Perform series evaluation
A8C6	A8D4	'ACS' function
A8CC	A8DA	'ASN' function
A907	A907	'ATN' function
A956	A95A	Constant series for 'ATN' evaluation
A989	A98D	'COS' function
A994	A998	'SIN' function
AA5C	AA48	Point &4B,&4C at 'coarse -PI/2'
AA60	AA4C	Point &4B ,&4C at adjustment to above
AA69	AA55	Point &4B,&4C at PI/2
AA6D	AA59	Constant: 'coarse -PI/2'
AA73	AASE	Constant: adjustment to 'coarse -PI/2'
AA77	AA63	Constant: PI/2
AA7C	AA68	Constant: PI/180 (for 'RAD')
AA81	AA6D	Constant: 180/PI (for 'DEG')

AA86	AA72	Constant series for ‘SIN’ evaluation
AAB4	AA91	‘EXP’ function
AB07	AAE4	Constant: e (‘EXP 1’)
AB0C	AAE9	Constant series for ‘EXP’ evaluation
AB56	AB33	‘ADVAL’ function
AB64	AB41	‘POINT’ function
AB92	AB6D	‘POS’ function
AB9B	AB76	‘VPOS’ function
ABAE	AB88	‘SGN’ function
ABCD	ABA8	‘LOG’ function
ABD6	ABB1	‘RAD’ function
ABE7	ABC2	‘DEG’ function
ABF0	ABCB	‘PI’ function
ABFB	ABD2	‘USR’ function
AC12	ABE9	‘EV AL’ function
AC55	AC2F	‘V AL’ function
AC9E	AC78	‘INT’ function
ACC4	AC9E	‘ASC’ function
ACD3	ACAD	‘INKEY’ function
ACDE	ACB8	‘EOF’ function
ACEA	ACC4	‘TRUE’ function
ACF7	ACD1	‘NOT’ function
AD08	ACE2	‘INSTR(’ function
AD8D	AD6A	‘ABS’ function
ADB5	AD8C	Unary ‘−’ function
AE1B	ADEC	Get <factor> or <string-factor> at PTRB
AE9C	AE6D	Get HEX number
AEE3	AEB4	‘TIME’ function
AEEF	AEC0	‘PAGE’ function
AEF9	AECA	‘FALSE’ function
AF00	AED1	‘LEN’ function
AF0B	AEDC	‘TOP’ function
AF26	AEF7	‘COUNT’ function
AF2B	AEFC	‘LOMEM’ function
AF32	AF03	‘HIMEM’ function
AF78	AF49	‘RND’ function
AF85	AF56	Load IntA from 00,X−03,X
AFB6	AF87	Spin random number generator
AFCE	AF9F	‘ERL’ function
AFD5	AFA6	‘ERR’ function
AFDC	AFAD	Perform INKEY
AFE8	AFB9	‘GET’ function
AFEE	AFBF	‘GET$’ function

AFFE	AFCC	‘LEFT$(’ function
B01D	AFEE	‘RIGHT$(’ function
B055	B026	‘INKEY$’ function
B05D	B02E	Set StrA to empty string
B068	B039	‘MID$(’ function
B0C3	B094	‘STR$’ function
B0F1	B0C2	‘STRING$(’ function
B141	B112	Search for FN/PROC not in list
B1C4	B195	‘FN’ function
B27C	B24D	Handle FN/PROC parameters
B33C	B30D	Push value and descriptor on STACK
B35B	B32C	Read value of variable
B3EE	B3BD	‘CHR$’ function
B3F6	B3C5	Set up ERL
B433	B402	BRK hander
B443	B433	Default BASIC error handling text
B461	B44C	‘SOUND’ statement
B49C	B472	‘ENVELOPE’ statement
B4CC	B4A0	‘WIDTH’ statement
B4E0	B4B1	Assign numeric variable
B53A	B50E	Print A as a character or token
8570	B545	Print A as 2-digit HEX number
B571	B558	Print A as a character (handling COUI
856A	B562	Print A as HEX number followed by sp
B58D	B577	Print selected LISTO formatting spaces
B5A0	B58A	‘LISTO’ command
B5B5	B59C	‘LIST’ command
B6AE	B695	‘NEXT’ statement
B7DF	B7C4	‘FOR’ statement
B8B4	B888	‘GOSUB’ statement
B8D5	B8B6	‘RETURN’ statement
B8EB	B8CC	‘GOTO’ statement
B903	B8E4	‘ON ERROR OFF’ statement
B911	B8F2	‘ON ERROR’ statement
B934	B915	‘ON’ statement
B9B8	B99A	Get line number, and find it in program
B9ED	B9CF	‘INPUT#’ statement
BA62	BA44	‘INPUT’ statement
BB00	BAE6	‘RESTORE’ statement
BB39	BBF1	‘READ’ statement
BBCC	EBB1	‘UNTIL’ statement
BBFF	BBE4	‘REPEAT’ statement
BC17	BBFC	Input string to StrA

BC1D	BC02	Input string to keyboard buffer
BC42	BC25	Print CRLF (newline)
BC4A	BC2D	Delete line in program
BCAA	BC8D	Insert line into program
BD29	BD11	'RUN' statement
BD38	BD20	Clear variablesv'stacks
BD52	BD3A	Reset stacks; RESTORE data pointer
BD69	BD51	Push FP A on ST ACK
BD96	BD7E	Pop real number from STACK
BDA8	BD90	Push IntA, FPA, or StrA on STACK
BDAC	BD94	Push IntA on ST ACK
BDCA	BDB2	Push StrA on ST ACK
BDE3	BDCB	Pop StrA from STACK
BDF4	BDDC	Discard string from STACK
BE04	BDEA	Pop IntA from STACK
BE17	BDFF	Discard integer (4 bytes) from STACK
BE23	BEOB	Pop integer from ST ACK to &37 —&3A
BE25	BEOD	Pop integer into page zero
BE46	BE2E	Allocate STACK space; check for 'No room'
BESC	BE44	Copy IntA into 0,X−3,X
BE6D	BESS	Add Y to pointer at &3D,&3E; Set Y=1
BE7A	BE62	Perform BASIC program load
BE88	BE6F	Test for 'Bad program'
−−−−	BEC2	'OSCLI' statement
BEEA	BEF3	'SAVE' statement
BF2D	BF24	'LOAD' statement
BF33	BF2A	'CHAIN' statement
BF39	BF30	'PTR' statement
BF4F	BF46	'EXT' function
BF50	BF47	'PTR' function
BF61	BF58	'BPUT' statement
BF78	BF6F	'BGET' function
−−−−	BF78	'OPENIN' function
BF81	BF7C	'OPENOUT' function
BF85	BF80	'OPENUP' function ('OPENIN' in BASIC 1)
BF9E	BF99	'CLOSE' statement
BFAE	BFA9	Get file handle at PTRA
BFCB	BFCF	Print text after 'JSR' to this routine
BFE6	BFE4	'REPORT'
−−−−	BFF9	Text: 'Roger'