Appendices

Appendix A – 68000 INSTRUCTION SET SUMMARY

A.1 Addressing modes

Six basic addressing modes in the 68000 give rise to 14 actual modes. The modes of addressing are shown in Fig.A.1, together with the appropriate assembler syntax.

MODE	SYNTAX
Implied	
Register	SR, CCR, USP, PC
Immediate	
Immediate	#n
Quick immediate	#b
Absolute	
Short	a16
Long	a32
Register Direct	
Data register	Dn
Address register direct	An
Register Indirect	
Address register	(An)
Postincrement	(An)+
Predecrement	-(An)
Address register with offset	d16(An)
Register with index and offset	d8(An,i)
Program Counter Relative	
Address register with offset	d16(PC)
Register with index and offset	d8(PC,i)

Notes:

b	= 3, 4 or 8 bits	i	= An or Dn
n	= 8,16, or 32 bits	An	= address register
d8	= 8 bit offset	Dn	= data register
d16	= 16 bit offset	PC	= current location
a16	= 16 bit address	SR	= status register
a32	= 32 bit address	CCR	= condition codes
		USP	= user stack ptr

Figure A.1 68000 addressing modes

A.2 Condition codes

There are three instructions (Bcc, DBcc, and Scc) which use a set of conditional tests. The tests are given 'one/two character' mnemonics and the full instruction mnemonic consists of the above names with 'cc' replaced by the test mnemonic (e.g., BHI, BF, DBEQ, SNE, and so on). Each test produces a true or false result depending on the state of given condition flags in the 68000 CCR register.

In the table below, the alternative mnemonics are given in parenthesis after the standard mnemonic.

Mnemonic	Test	Interpretation
T	1	true (always)
F	0	false (always)
HI	not(C).not(Z)	higher (unsigned)
LS	C+Z	less than or same (unsigned)
CC (HS)	not(C)	carry clear (unsigned)
CS (LO)	C	carry set (unsigned)
NE	not(Z)	not equal
EQ	Z	equal
VC	not(V)	overflow clear
VS	V	overflow set
PL	not(N)	plus
MI	N	minus
GE	not(N xor V)	greater than or equal (signed)
LT	N xor V	less than (signed)
GT	not(Z+(N xor V))	greater than
LE	Z+(N xor V)	less than or equal

A.3 68000 instruction set summary

In Fig.A.2 (below) the instruction set of the 68000 MPU is given in alphabetic order. The effect of each instruction on the CCR flags is supplied, together with an indication of whether or not the instruction is privileged (i.e., can only be executed while the 68000 is in supervisor mode). Within the condition code list, the following key is used:

x : flag is affected
u : flag is undefined
- : flag is unaffected
0 : flag is reset to zero
1 : flag is set to one

The privileged instruction column (P) uses the following key:

n : not a privileged instruction
y : privileged instruction
? : privileged under certain conditions

If a '?' does appear in the 'P' column, reference should be made to Chap.2 in order to determine which special cases can occur.

		X N Z V C	P
ABCD	Add decimal with extend	x u x u x	n
ADD	Add	x x x x x	n
	(When destination is 'An')	- - - - -	n
ADDQ	Add quick	x x x x x	n
ADDX	Add with extend	x x x x x	n
AND	Logical AND	- x x 0 0	?
ASL	Arithmetic shift left	x x x x x	n
ASR	Arithmetic shift right	x x x x x	n
Bcc	Branch conditionally	- - - - -	n
BCHG	Bit test and change	- - x - -	n
BCLR	Bit test and clear	- - x - -	n
BRA	Branch always	- - - - -	n
BSET	Bit test and set	- - x - -	n
BSR	Branch to subroutine	- - - - -	n
BTST	Bit test	- - x - -	n
CHK	Check reg. against bounds	- x u u u	n
CLR	Clear operand	- 0 1 0 0	n
CMP	Compare	- x x x x	n
CMPM	Compare memory	- x x x x	n
DBcc	Dec. and branch cond.	- - - - -	n
DBRA	Decrement and branch always	- - - - -	n
DIVS	Signed divide	- x x x 0	n
DIVU	Unsigned divide	- x x x 0	n
EOR	Exclusive OR	- x x 0 0	?
EXG	Exchange registers	- - - - -	n
EXT	Sign extend	- x x 0 0	n
JMP	Jump	- - - - -	n
JSR	Jump to subroutine	- - - - -	n
LEA	Load effective address	- - - - -	n
LINK	Link stack	- - - - -	n
LSL	Logical shift left	x x x 0 x	n
LSR	Logical shift right	x x x 0 x	n
MOVE	Move	- x x 0 0	n
	(When dest. is 'An')	- - - - -	n
	(When dest. is 'CCR')	x x x x x	n
	(When src. is 'SR')	- - - - -	n
	(When dest. is 'SR')	x x x x x	y
	(When 'USP' used)	- - - - -	y
MOVEM	Move multiple registers	- - - - -	n

		X	N	Z	V	C	P
MOVEP	Move peripheral data	-	-	-	-	-	n
MOVEQ	Move quick	-	x	x	0	0	n
MULS	Signed multiply	-	x	x	0	0	n
MULU	Unsigned multiply	-	x	x	0	0	n
NBCD	Negate decimal with extend	x	u	x	u	x	n
NEG	Negate	x	x	x	x	x	n
NEGX	Negate with extend	x	x	x	x	x	n
NOP	No operation	-	-	-	-	-	n
NOT	One's complement	-	x	x	0	0	n
OR	Logical OR	-	x	x	0	0	?
PEA	Push effective address	-	-	-	-	-	n
RESET	Reset external devices	-	-	-	-	-	y
ROL	Rotate left	-	x	x	0	x	n
ROR	Rotate right	-	x	x	0	x	n
ROXL	Rotate left through extend	x	x	x	0	x	n
ROXR	Rotate right through extend	x	x	x	0	x	n
RTE	Return from exception	x	x	x	x	x	y
RTR	Return and restore CCR	x	x	x	x	x	n
RTS	Return from subroutine	-	-	-	-	-	n
SBCD	Subtract decimal with extend	x	u	x	u	x	n
Scc	Set conditional	-	-	-	-	-	n
STOP	Stop	x	x	x	x	x	y
SUB	Subtract	x	x	x	x	x	n
	(When destination is 'An')	-	-	-	-	-	n
SUBQ	Subtract quick	x	x	x	x	x	n
SUBX	Subtract with extend	x	x	x	x	x	n
SWAP	Swap data register halves	-	x	x	0	0	n
TAS	Test and set bit 7	-	x	x	0	0	n
TRAP	Trap	-	-	-	-	-	n
TRAPV	Trap on overflow	-	-	-	-	-	n
TST	Test	-	x	x	0	0	n
UNLK	Unlink	-	-	-	-	-	n

Figure A.2 68000 instruction set summary

Appendix B – QL SYSTEM CALL SUMMARY

Given here is a list of all the QDOS system 'TRAP#n' calls, and 'vectored' utility calls, described in detail in Chapters 4 to 7. The trap code for trap calls #1 to #3 are passed in register 'D0'. There are a number of other calls and utilities available which are of use when, for example, writing device drivers for external I/O (on extension cards). These have not been previously discussed and are not listed below, as they are outside the scope of this text.

	TRAP #1 MACHINE RESOURCE MANAGEMENT		
MNEMONIC	CODE (hex.)	CODE (den.)	DESCRIPTION
MT.INF	0	0	Get system information
MT.CJOB	1	1	Create a job in TPA
MT.JINF	2	2	Get job information
MT.RJOB	4	4	Remove inactive job from TPA
MT.FRJOB	5	5	Force remove job(s) from TPA
MT.FREE	6	6	Length of largest space in TPA
MT.TRAPV	7	7	Set job trap vector pointer
MT.SUSJB	8	8	Suspend a job
MT.RELJB	9	9	Release a job & re-schedule
MT.ACTIV	A	10	Activate a job
MT.PRIOR	B	11	Change job priority
MT.ALRES	E	14	Allocate resident procedure area
MT.RERES	F	15	Release resident procedure area
MT.DMODE	10	16	Set/read display mode
MT.IPCOM	11	17	IPC command (kbd row scan, sound)
MT.BAUD	12	18	Set baud rate
MT.RCLCK	13	19	Read real-time clock
MT.SCLCK	14	20	Set real-time clock
MT.ACLCK	15	21	Adjust real-time clock
MT.ALCHP	18	24	Allocate common heap area
MT.RECHP	19	25	Release common heap area

TRAP #2 INPUT/OUTPUT ALLOCATION			
MNEMONIC	CODE (hex.)	CODE (den.)	DESCRIPTION
IO.OPEN	1	1	Open a channel
IO.CLOSE	2	2	Close a channel
IO.FORMT	3	3	Format a sectored medium
IO.DELET	4	4	Delete a file

TRAP #3 INPUT/OUTPUT OPERATIONS			
MNEMONIC	CODE (hex.)	CODE (den.)	DESCRIPTION
IO.PEND	0	0	Check for pending input
IO.FBYTE	1	1	Fetch a byte
IO.FLINE	2	2	Fetch a line (terminator LF)
IO.FSTRG	3	3	Fetch a string of bytes
IO.EDLIN	4	4	Edit a line (console only)
IO.SBYTE	5	5	Send a byte
IO.SSTRG	7	7	Send a string of bytes
IO.EXTOP	9	9	Call an extended operation
SD.PXENQ	A	10	Return window size & cursor position in pixel coords.
SD.CHENQ	B	11	Return window size & cursor position in character coords.
SD.BORDR	C	12	Set border width & colour
SD.WDEF	D	13	Define window
SD.CURE	E	14	Enable cursor
SD.CURS	F	15	Suppress cursor
SD.POS	10	16	Move cursor absolute (char.)
SD.TAB	11	17	Tabulate
SD.NL	12	18	Newline
SD.PCOL	13	19	Cursor back
SD.NCOL	14	20	Cursor forward
SD.PROW	15	21	Cursor up
SD.NROW	16	22	Cursor down
SD.PIXP	17	23	Move cursor absolute (pixel)
SD.SCROL	18	24	Scroll entire window
SD.SCRTP	19	25	Scroll top of window
SD.SCRBT	1A	26	Scroll bottom of window
SD.PAN	1B	27	Pan entire window
SD.PANLN	1E	30	Pan cursor line
SD.PANRT	1F	31	Pan RHS of cursor line
SD.CLEAR	20	32	Clear entire window
SD.CLRTP	21	33	Clear top of window
SD.CLRBT	22	34	Clear bottom of window
SD.CLRLN	23	35	Clear cursor line

SD.CLRRT	24	36	Clear RHS of cursor line
SD.FONT	25	37	Set/reset character font
SD.RECOL	26	38	Re-colour a window
SD.SETPA	27	39	Set paper colour
SD.SETST	28	40	Set strip colour
SD.SETIN	29	41	Set ink colour
SD.SETFL	2A	42	Set/reset flash
SD.SETUL	2B	43	Set/reset underscore
SD.SETMD	2C	44	Set writing/plotting mode
SD.SETSZ	2D	45	Set character size
SD.FILL	2E	46	Fill rectangle
SD.POINT	30	48	Plot a point
SD.LINE	31	49	Plot a line
SD.ARC	32	50	Plot an arc
SD.ELIPS	33	51	Plot an ellipse
SD.SCALE	34	52	Set scale
SD.FLOOD	35	53	Set/reset area flood
SD.GCUR	36	54	Set graphic cursor position
FS.CHECK	40	64	Check pending file operations
FS.FLUSH	41	65	Flush file buffers
FS.POSAB	42	66	Set file pointer absolute
FS.POSRE	43	67	Set file pointer relative
FS.MDINF	45	69	Get medium information
FS.HEADS	46	70	Set file header
FS.HEADR	47	71	Read file header
FS.LOAD	48	72	Load a file
FS.SAVE	49	73	Save a file

VECTORED
UTILITY ROUTINES

MNEMONIC	VECTOR (hex.)	VECTOR (den.)	DESCRIPTION
UT.WINDW	C4	196	Set window using name
UT.CON	C6	198	Set up console window
UT.SCR	C8	200	Set up screen window
UT.ERRO	CA	202	Write error message to channel 0
UT.ERR	CC	204	Write error message to channel n
UT.MINT	CE	206	Convert integer to ASCII and write it to channel n.
UT.MTEXT	D0	208	Send message to channel n
UT.CSTR	E6	230	Compare two strings
CN.DATE	EC	236	Get date and time
CN.DAY	EE	238	Get day of week
CN.FTOD	F0	240	Convert floating point to ASCII
CN.ITOD	F2	242	Convert integer to ASCII
CN.ITOBB	F4	244	Convert byte to ASCII
CN.ITOBW	F6	246	Convert word to ASCII
CN.ITOBL	F8	248	Convert long-word to ASCII
CN.ITOHB	FA	250	Convert byte to hex. ASCII

CN.ITOHW	FC	252	Convert word to hex. ASCII
CN.ITOHL	FE	254	Convert long-word to hex. ASCII
CN.DTOF	100	256	Convert ASCII to floating point
CN.DTOI	102	258	Convert ASCII to integer
CN.BTOIB	104	260	Convert ASCII to byte
CN.BTOIW	106	262	Convert ASCII to word
CN.BTOIL	108	264	Convert ASCII to long-word
CN.HTOIB	10A	266	Convert hex. ASCII to byte
CN.HTOIW	10C	268	Convert hex. ASCII to word
CN.HTOIL	10E	270	Convert hex. ASCII to long-word
RI.EXEC	11C	284	Execute single arithmetic op.
RI.EXECB	11E	286	Execute list of arithmetic ops.

The four Microdrive support utilities mentioned at the end of Chapter 3 are not included here, as they are outside the scope of this book. They would normally be used for direct sector reading, writing, and verification.

Appendix C – 'QDOS' SYSTEM ERROR CODES

The QDOS system recognises 21 standard error conditions. These error conditions may occur (and be reported) either from within a SuperBASIC program or an assembly language program. In the latter case the error code is returned in register 'D0', as shown in Chapters 4 to 7 of Part 2 (QL System Procedures). All error codes are 'long words' (i.e., 32 bits). The system errors are:

MNEMONIC	VALUE	DESCRIPTION
ERR.NC	-1	Operation not complete
ERR.NJ	-2	Not a valid job
ERR.OM	-3	Out of memory
ERR.OR	-4	Out of range
ERR.BO	-5	Buffer overflow
ERR.NO	-6	Channel not open
ERR.NF	-7	File, device, variable or procedure not found
ERR.EX	-8	File already exists
ERR.IU	-9	File (or device) in use
ERR.EF	-10	End of file
ERR.DF	-11	Drive full
ERR.BN	-12	Bad device or procedure name
ERR.TE	-13	Transmission error
ERR.FF	-14	Format failed
ERR.BP	-15	Bad parameter
ERR.FE	-16	File error
ERR.XP	-17	Expression error
ERR.OV	-18	Arithmetic overflow
ERR.NI	-19	Not implemented (yet)
ERR.RO	-20	Read only
ERR.BL	-21	Bad line syntax (SuperBASIC)

Note that all the error code values are small negative integers. This structure enables standard error codes to be distinguished clearly from pointers to specific device-driver error messages, as the latter are passed as (pointer.to.message - $8000).

Appendix D – EDITOR/ ASSEMBLER QUICK REFERENCE GUIDE

The editor and assembler packages are discussed in Chapters 13 and 14. Given here are quick reference guides for their use.

EDITOR REFERENCE GUIDE

a) Top level commands:

```
^E  -  Execute extended command:
       -  D  -  Delete block (marker to cursor inclusive)
       -  F  -  Find text string
       -  I  -  Include external file (at cursor)
       -  M  -  Move to line absolute
^F  -  Finish - return to SuperBASIC
^H  -  Give help on cursor control and deletion commands
^L  -  Load a file in from Microdrive
^M  -  Set current cursor line as marker line
^S  -  Save editor buffer on to Microdrive
```

b) Cursor control commands:

KEY	NORMAL	SHIFT	ALTMODE
up	line	page	start of text
down	line	page	end of text
left	character	word	start of line
right	character	word	end of line

c) Text deletion commands:

KEY	CTRL
up/down	delete current line
left	delete char/gap left
right	delete cursor char/gap

ASSEMBLER/LOCATOR REFERENCE GUIDE

a) Comments:

Must be preceded by a semi-colon (;)

b) Labels:

Must be followed by a colon (:)

c) Directives:

i)	*EJECT	-	Force a new page
ii)	*HEADING	-	Create new heading and new page
iii)	*LIST <on/off>	-	Switch listing file on/off
iv)	*NUMBER <on/off>	-	Switch line numbers on/off
v)	*INCLUDE <file>	-	Include external source file

d) Pseudo-operators:

i)	EQU (=)	-	Static equate
ii)	QRY	-	Dynamic equate
iii)	ORG	-	Set program counter
iv)	ALIGN	-	Align to word boundary
v)	COND <expr>	-	Conditional assembly
	ELSE		
	ENDC		

e) Expression operators:

+	add	SHR	shift right
-	subtract	SHL	shift left
*	multiply	OR	logical 'or'
/	divide	AND	logical 'and'
		NOT	ones's complement

INDEX