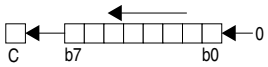
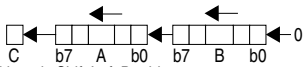
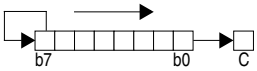
Table A-1. Instruction Set Summary (Sheet 1 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
ABA	(A) + (B) ⇒ A Add Accumulators A and B	INH	18 06	OO	OO	– – Δ –	Δ Δ Δ Δ
ABX	(B) + (X) ⇒ X *Translates to* LEAX B,X	IDX	1A E5	Pf	PP[1]	– – – –	– – – –
ABY	(B) + (Y) ⇒ Y *Translates to* LEAY B,Y	IDX	19 ED	Pf	PP[1]	– – – –	– – – –
ADCA *#opr8i*	(A) + (M) + C ⇒ A Add with Carry to A	IMM	89 ii	P	P	– – Δ –	Δ Δ Δ Δ
ADCA *opr8a*		DIR	99 dd	rPf	rfP		
ADCA *opr16a*		EXT	B9 hh ll	rPO	rOP		
ADCA *oprx0_xysp*		IDX	A9 xb	rPf	rfP		
ADCA *oprx9,xysp*		IDX1	A9 xb ff	rPO	rPO		
ADCA *oprx16,xysp*		IDX2	A9 xb ee ff	frPP	frPP		
ADCA [D,*xysp*]		[D,IDX]	A9 xb	fIfrPf	fIfrfP		
ADCA [*oprx16,xysp*]		[IDX2]	A9 xb ee ff	fIPrPf	fIPrfP		
ADCB *#opr8i*	(B) + (M) + C ⇒ B Add with Carry to B	IMM	C9 ii	P	P	– – Δ –	Δ Δ Δ Δ
ADCB *opr8a*		DIR	D9 dd	rPf	rfP		
ADCB *opr16a*		EXT	F9 hh ll	rPO	rOP		
ADCB *oprx0_xysp*		IDX	E9 xb	rPf	rfP		
ADCB *oprx9,xysp*		IDX1	E9 xb ff	rPO	rPO		
ADCB *oprx16,xysp*		IDX2	E9 xb ee ff	frPP	frPP		
ADCB [D,*xysp*]		[D,IDX]	E9 xb	fIfrPf	fIfrfP		
ADCB [*oprx16,xysp*]		[IDX2]	E9 xb ee ff	fIPrPf	fIPrfP		
ADDA *#opr8i*	(A) + (M) ⇒ A Add without Carry to A	IMM	8B ii	P	P	– – Δ –	Δ Δ Δ Δ
ADDA *opr8a*		DIR	9B dd	rPf	rfP		
ADDA *opr16a*		EXT	BB hh ll	rPO	rOP		
ADDA *oprx0_xysp*		IDX	AB xb	rPf	rfP		
ADDA *oprx9,xysp*		IDX1	AB xb ff	rPO	rPO		
ADDA *oprx16,xysp*		IDX2	AB xb ee ff	frPP	frPP		
ADDA [D,*xysp*]		[D,IDX]	AB xb	fIfrPf	fIfrfP		
ADDA [*oprx16,xysp*]		[IDX2]	AB xb ee ff	fIPrPf	fIPrfP		
ADDB *#opr8i*	(B) + (M) ⇒ B Add without Carry to B	IMM	CB ii	P	P	– – Δ –	Δ Δ Δ Δ
ADDB *opr8a*		DIR	DB dd	rPf	rfP		
ADDB *opr16a*		EXT	FB hh ll	rPO	rOP		
ADDB *oprx0_xysp*		IDX	EB xb	rPf	rfP		
ADDB *oprx9,xysp*		IDX1	EB xb ff	rPO	rPO		
ADDB *oprx16,xysp*		IDX2	EB xb ee ff	frPP	frPP		
ADDB [D,*xysp*]		[D,IDX]	EB xb	fIfrPf	fIfrfP		
ADDB [*oprx16,xysp*]		[IDX2]	EB xb ee ff	fIPrPf	fIPrfP		
ADDD *#opr16i*	(A:B) + (M:M+1) ⇒ A:B Add 16-Bit to D (A:B)	IMM	C3 jj kk	PO	OP	– – – –	Δ Δ Δ Δ
ADDD *opr8a*		DIR	D3 dd	RPf	RfP		
ADDD *opr16a*		EXT	F3 hh ll	RPO	ROP		
ADDD *oprx0_xysp*		IDX	E3 xb	RPf	RfP		
ADDD *oprx9,xysp*		IDX1	E3 xb ff	RPO	RPO		
ADDD *oprx16,xysp*		IDX2	E3 xb ee ff	fRPP	fRPP		
ADDD [D,*xysp*]		[D,IDX]	E3 xb	fIfRPf	fIfRfP		
ADDD [*oprx16,xysp*]		[IDX2]	E3 xb ee ff	fIPRPf	fIPRfP		
ANDA *#opr8i*	(A) • (M) ⇒ A Logical AND A with Memory	IMM	84 ii	P	P	– – – –	Δ Δ 0 –
ANDA *opr8a*		DIR	94 dd	rPf	rfP		
ANDA *opr16a*		EXT	B4 hh ll	rPO	rOP		
ANDA *oprx0_xysp*		IDX	A4 xb	rPf	rfP		
ANDA *oprx9,xysp*		IDX1	A4 xb ff	rPO	rPO		
ANDA *oprx16,xysp*		IDX2	A4 xb ee ff	frPP	frPP		
ANDA [D,*xysp*]		[D,IDX]	A4 xb	fIfrPf	fIfrfP		
ANDA [*oprx16,xysp*]		[IDX2]	A4 xb ee ff	fIPrPf	fIPrfP		
ANDB *#opr8i*	(B) • (M) ⇒ B Logical AND B with Memory	IMM	C4 ii	P	P	– – – –	Δ Δ 0 –
ANDB *opr8a*		DIR	D4 dd	rPf	rfP		
ANDB *opr16a*		EXT	F4 hh ll	rPO	rOP		
ANDB *oprx0_xysp*		IDX	E4 xb	rPf	rfP		
ANDB *oprx9,xysp*		IDX1	E4 xb ff	rPO	rPO		
ANDB *oprx16,xysp*		IDX2	E4 xb ee ff	frPP	frPP		
ANDB [D,*xysp*]		[D,IDX]	E4 xb	fIfrPf	fIfrfP		
ANDB [*oprx16,xysp*]		[IDX2]	E4 xb ee ff	fIPrPf	fIPrfP		
ANDCC *#opr8i*	(CCR) • (M) ⇒ CCR Logical AND CCR with Memory	IMM	10 ii	P	P	⇓ ⇓ ⇓ ⇓	⇓ ⇓ ⇓ ⇓

Note 1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.

Table A-1. Instruction Set Summary (Sheet 2 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
ASL *opr16a* ASL *oprx0_xysp* ASL *oprx9,xysp* ASL *oprx16,xysp* ASL [D,*xysp*] ASL [*oprx16,xysp*] ASLA ASLB	 Arithmetic Shift Left Arithmetic Shift Left Accumulator A Arithmetic Shift Left Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	78 hh ll 68 xb 68 xb ff 68 xb ee ff 68 xb 68 xb ee ff 48 58	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	ΔΔΔΔ
ASLD	 Arithmetic Shift Left Double	INH	59	O	O	----	ΔΔΔΔ
ASR *opr16a* ASR *oprx0_xysp* ASR *oprx9,xysp* ASR *oprx16,xysp* ASR [D,*xysp*] ASR [*oprx16,xysp*] ASRA ASRB	 Arithmetic Shift Right Arithmetic Shift Right Accumulator A Arithmetic Shift Right Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	77 hh ll 67 xb 67 xb ff 67 xb ee ff 67 xb 67 xb ee ff 47 57	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	ΔΔΔΔ
BCC *rel8*	Branch if Carry Clear (if C = 0)	REL	24 rr	PPP/P[1]	PPP/P[1]	----	----
BCLR *opr8a*, *msk8* BCLR *opr16a*, *msk8* BCLR *oprx0_xysp*, *msk8* BCLR *oprx9,xysp*, *msk8* BCLR *oprx16,xysp*, *msk8*	(M) • ($\overline{mm}$) ⇒ M Clear Bit(s) in Memory	DIR EXT IDX IDX1 IDX2	4D dd mm 1D hh ll mm 0D xb mm 0D xb ff mm 0D xb ee ff mm	rPwO rPwP rPwO rPwP frPwPO	rPOw rPPw rPOw rPwP frPwOP	----	ΔΔ0–
BCS *rel8*	Branch if Carry Set (if C = 1)	REL	25 rr	PPP/P[1]	PPP/P[1]	----	----
BEQ *rel8*	Branch if Equal (if Z = 1)	REL	27 rr	PPP/P[1]	PPP/P[1]	----	----
BGE *rel8*	Branch if Greater Than or Equal (if N ⊕ V = 0) (signed)	REL	2C rr	PPP/P[1]	PPP/P[1]	----	----
BGND	Place CPU in Background Mode see *CPU12 Reference Manual*	INH	00	VfPPP	VfPPP	----	----
BGT *rel8*	Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	2E rr	PPP/P[1]	PPP/P[1]	----	----
BHI *rel8*	Branch if Higher (if C + Z = 0) (unsigned)	REL	22 rr	PPP/P[1]	PPP/P[1]	----	----
BHS *rel8*	Branch if Higher or Same (if C = 0) (unsigned) same function as BCC	REL	24 rr	PPP/P[1]	PPP/P[1]	----	----
BITA #*opr8i* BITA *opr8a* BITA *opr16a* BITA *oprx0_xysp* BITA *oprx9,xysp* BITA *oprx16,xysp* BITA [D,*xysp*] BITA [*oprx16,xysp*]	(A) • (M) Logical AND A with Memory Does not change Accumulator or Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	85 ii 95 dd B5 hh ll A5 xb A5 xb ff A5 xb ee ff A5 xb A5 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP rPO frPP fIfrfP fIPrfP	----	ΔΔ0–
BITB #*opr8i* BITB *opr8a* BITB *opr16a* BITB *oprx0_xysp* BITB *oprx9,xysp* BITB *oprx16,xysp* BITB [D,*xysp*] BITB [*oprx16,xysp*]	(B) • (M) Logical AND B with Memory Does not change Accumulator or Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C5 ii D5 dd F5 hh ll E5 xb E5 xb ff E5 xb ee ff E5 xb E5 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP rPO frPP fIfrfP fIPrfP	----	ΔΔ0–
BLE *rel8*	Branch if Less Than or Equal (if Z + (N ⊕ V) = 1) (signed)	REL	2F rr	PPP/P[1]	PPP/P[1]	----	----
BLO *rel8*	Branch if Lower (if C = 1) (unsigned) same function as BCS	REL	25 rr	PPP/P[1]	PPP/P[1]	----	----

Note 1. PPP/P indicates this instruction takes three cycles to refill the instruction queue if the branch is taken and one program fetch cycle if the branch is not taken.

Table A-1. Instruction Set Summary (Sheet 3 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
BLS *rel8*	Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	23 rr	PPP/P[1]	PPP/P[1]	----	----
BLT *rel8*	Branch if Less Than (if N ⊕ V = 1) (signed)	REL	2D rr	PPP/P[1]	PPP/P[1]	----	----
BMI *rel8*	Branch if Minus (if N = 1)	REL	2B rr	PPP/P[1]	PPP/P[1]	----	----
BNE *rel8*	Branch if Not Equal (if Z = 0)	REL	26 rr	PPP/P[1]	PPP/P[1]	----	----
BPL *rel8*	Branch if Plus (if N = 0)	REL	2A rr	PPP/P[1]	PPP/P[1]	----	----
BRA *rel8*	Branch Always (if 1 = 1)	REL	20 rr	PPP	PPP	----	----
BRCLR *opr8a, msk8, rel8* BRCLR *opr16a, msk8, rel8* BRCLR *oprx0_xysp, msk8, rel8* BRCLR *oprx9,xysp, msk8, rel8* BRCLR *oprx16,xysp, msk8, rel8*	Branch if (M) • (mm) = 0 (if All Selected Bit(s) Clear)	DIR EXT IDX IDX1 IDX2	4F dd mm rr 1F hh ll mm rr 0F xb mm rr 0F xb ff mm rr 0F xb ee ff mm rr	rPPP rfPPP rPPP rfPPP PrfPPP	rPPP rfPPP rPPP rffPPP frPffPPP	----	----
BRN *rel8*	Branch Never (if 1 = 0)	REL	21 rr	P	P	----	----
BRSET *opr8, msk8, rel8* BRSET *opr16a, msk8, rel8* BRSET *oprx0_xysp, msk8, rel8* BRSET *oprx9,xysp, msk8, rel8* BRSET *oprx16,xysp, msk8, rel8*	Branch if ($\overline{M}$) • (mm) = 0 (if All Selected Bit(s) Set)	DIR EXT IDX IDX1 IDX2	4E dd mm rr 1E hh ll mm rr 0E xb mm rr 0E xb ff mm rr 0E xb ee ff mm rr	rPPP rfPPP rPPP rfPPP PrfPPP	rPPP rfPPP rPPP rffPPP frPffPPP	----	----
BSET *opr8, msk8* BSET *opr16a, msk8* BSET *oprx0_xysp, msk8* BSET *oprx9,xysp, msk8* BSET *oprx16,xysp, msk8*	(M) + (mm) ⇒ M Set Bit(s) in Memory	DIR EXT IDX IDX1 IDX2	4C dd mm 1C hh ll mm 0C xb mm 0C xb ff mm 0C xb ee ff mm	rPwO rPwP rPwO rPwP frPwPO	rPOw rPPw rPOw rPwP frPwOP	----	Δ Δ 0 –
BSR *rel8*	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$ Subroutine address ⇒ PC Branch to Subroutine	REL	07 rr	SPPP	PPPS	----	----
BVC *rel8*	Branch if Overflow Bit Clear (if V = 0)	REL	28 rr	PPP/P[1]	PPP/P[1]	----	----
BVS *rel8*	Branch if Overflow Bit Set (if V = 1)	REL	29 rr	PPP/P[1]	PPP/P[1]	----	----
CALL *opr16a, page* CALL *oprx0_xysp, page* CALL *oprx9,xysp, page* CALL *oprx16,xysp, page* CALL [D,*xysp*] CALL [*oprx16, xysp*]	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$ (SP) – 1 ⇒ SP; (PPG) ⇒ $M_{(SP)}$; pg ⇒ PPAGE register; Program address ⇒ PC Call subroutine in extended memory (Program may be located on another expansion memory page.) Indirect modes get program address and new pg value based on pointer.	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	4A hh ll pg 4B xb pg 4B xb ff pg 4B xb ee ff pg 4B xb 4B xb ee ff	gnSsPPP gnSsPPP gnSsPPP fgnSsPPP fIignSsPPP fIignSsPPP	gnfSsPPP gnfSsPPP gnfSsPPP fgnfSsPPP fIignSsPPP fIignSsPPP	----	----
CBA	(A) – (B) Compare 8-Bit Accumulators	INH	18 17	OO	OO	----	Δ Δ Δ Δ
CLC	0 ⇒ C *Translates to* ANDCC #$FE	IMM	10 FE	P	P	----	---0
CLI	0 ⇒ I *Translates to* ANDCC #$EF (enables I-bit interrupts)	IMM	10 EF	P	P	---0	----
CLR *opr16a* CLR *oprx0_xysp* CLR *oprx9,xysp* CLR *oprx16,xysp* CLR [D,*xysp*] CLR [*oprx16,xysp*] CLRA CLRB	0 ⇒ M Clear Memory Location 0 ⇒ A Clear Accumulator A 0 ⇒ B Clear Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	79 hh ll 69 xb 69 xb ff 69 xb ee ff 69 xb 69 xb ee ff 87 C7	PwO Pw PwO PwP PIfw PIPw O O	wOP Pw PwO PwP PIfPw PIPPw O O	----	0 1 0 0
CLV	0 ⇒ V *Translates to* ANDCC #$FD	IMM	10 FD	P	P	----	--0–

Note 1. PPP/P indicates this instruction takes three cycles to refill the instruction queue if the branch is taken and one program fetch cycle if the branch is not taken.

CMPA #*opr8i*
CMPA *opr8a*
CMPA *opr16a*
CMPA *oprx0_xysp*
CMPA *oprx9,xysp*
CMPA *oprx16,xysp*
CMPA [D,*xysp*]
CMPA [*oprx16,xysp*]
	(A) – (M)
	Compare Accumulator A with Memory
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			81 ii
			91 dd
			B1 hh ll
			A1 xb
			A1 xb ff
			A1 xb ee ff
			A1 xb
			A1 xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rfP
					rOP
					rfP
					rPO
					frPP
					fIfrfP
					fIPrfP
						----	Δ Δ Δ Δ

Table A-1. Instruction Set Summary (Sheet 4 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C

CMPB #*opr8i*
CMPB *opr8a*
CMPB *opr16a*
CMPB *oprx0_xysp*
CMPB *oprx9,xysp*
CMPB *oprx16,xysp*
CMPB [D,*xysp*]
CMPB [*oprx16,xysp*]
	(B) – (M)
	Compare Accumulator B with Memory
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			C1 ii
			D1 dd
			F1 hh ll
			E1 xb
			E1 xb ff
			E1 xb ee ff
			E1 xb
			E1 xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rfP
					rOP
					rfP
					rPO
					frPP
					fIfrfP
					fIPrfP
						----	Δ Δ Δ Δ

COM *opr16a* COM *oprx0_xysp* COM *oprx9,xysp* COM *oprx16,xysp* COM [D,*xysp*] COM [*oprx16,xysp*] COMA COMB	($\overline{M}$) ⇒ M *equivalent to* $FF – (M) ⇒ M 1's Complement Memory Location ($\overline{A}$) ⇒ A Complement Accumulator A ($\overline{B}$) ⇒ B Complement Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	71 hh ll 61 xb 61 xb ff 61 xb ee ff 61 xb 61 xb ee ff 41 51	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	Δ Δ 0 1
CPD #*opr16i* CPD *opr8a* CPD *opr16a* CPD *oprx0_xysp* CPD *oprx9,xysp* CPD *oprx16,xysp* CPD [D,*xysp*] CPD [*oprx16,xysp*]	(A:B) – (M:M+1) Compare D to Memory (16-Bit)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8C jj kk 9C dd BC hh ll AC xb AC xb ff AC xb ee ff AC xb AC xb ee ff	PO RPf RPO RPf RPO fRPP fIfRPf fIPRPf	OP RfP ROP RfP RPO fRPP fIfRfP fIPRfP	----	Δ Δ Δ Δ
CPS #*opr16i* CPS *opr8a* CPS *opr16a* CPS *oprx0_xysp* CPS *oprx9,xysp* CPS *oprx16,xysp* CPS [D,*xysp*] CPS [*oprx16,xysp*]	(SP) – (M:M+1) Compare SP to Memory (16-Bit)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8F jj kk 9F dd BF hh ll AF xb AF xb ff AF xb ee ff AF xb AF xb ee ff	PO RPf RPO RPf RPO fRPP fIfRPf fIPRPf	OP RfP ROP RfP RPO fRPP fIfRfP fIPRfP	----	Δ Δ Δ Δ
CPX #*opr16i* CPX *opr8a* CPX *opr16a* CPX *oprx0_xysp* CPX *oprx9,xysp* CPX *oprx16,xysp* CPX [D,*xysp*] CPX [*oprx16,xysp*]	(X) – (M:M+1) Compare X to Memory (16-Bit)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8E jj kk 9E dd BE hh ll AE xb AE xb ff AE xb ee ff AE xb AE xb ee ff	PO RPf RPO RPf RPO fRPP fIfRPf fIPRPf	OP RfP ROP RfP RPO fRPP fIfRfP fIPRfP	----	Δ Δ Δ Δ
CPY #*opr16i* CPY *opr8a* CPY *opr16a* CPY *oprx0_xysp* CPY *oprx9,xysp* CPY *oprx16,xysp* CPY [D,*xysp*] CPY [*oprx16,xysp*]	(Y) – (M:M+1) Compare Y to Memory (16-Bit)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8D jj kk 9D dd BD hh ll AD xb AD xb ff AD xb ee ff AD xb AD xb ee ff	PO RPf RPO RPf RPO fRPP fIfRPf fIPRPf	OP RfP ROP RfP RPO fRPP fIfRfP fIPRfP	----	Δ Δ Δ Δ
DAA	Adjust Sum to BCD Decimal Adjust Accumulator A	INH	18 07	OfO	OfO	----	Δ Δ ? Δ
DBEQ *abdxys, rel9*	(cntr) – 1⇒ cntr if (cntr) = 0, then Branch else Continue to next instruction Decrement Counter and Branch if = 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP	----	----
DBNE *abdxys, rel9*	(cntr) – 1 ⇒ cntr If (cntr) not = 0, then Branch; else Continue to next instruction Decrement Counter and Branch if ≠ 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP	----	----

Table A-1. Instruction Set Summary (Sheet 5 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
DEC *opr16a*	(M) – $01 ⇒ M	EXT	73 hh ll	rPwO	rOPw	----	ΔΔΔ–
DEC *oprx0_xysp*	Decrement Memory Location	IDX	63 xb	rPw	rPw		
DEC *oprx9,xysp*		IDX1	63 xb ff	rPwO	rPOw		
DEC *oprx16,xysp*		IDX2	63 xb ee ff	frPwP	frPPw		
DEC [D,*xysp*]		[D,IDX]	63 xb	fIfrPw	fIfrPw		
DEC [*oprx16,xysp*]		[IDX2]	63 xb ee ff	fIPrPw	fIPrPw		
DECA	(A) – $01 ⇒ A Decrement A	INH	43	O	O		
DECB	(B) – $01 ⇒ B Decrement B	INH	53	O	O		
DES	(SP) – $0001 ⇒ SP *Translates to* LEAS –1,SP	IDX	1B 9F	Pf	PP[1]	----	----
DEX	(X) – $0001 ⇒ X Decrement Index Register X	INH	09	O	O	----	–Δ––
DEY	(Y) – $0001 ⇒ Y Decrement Index Register Y	INH	03	O	O	----	–Δ––
EDIV	(Y:D) ÷ (X) ⇒ Y Remainder ⇒ D 32 by 16 Bit ⇒ 16 Bit Divide (unsigned)	INH	11	fffffffffffO	fffffffffffO	----	ΔΔΔΔ
EDIVS	(Y:D) ÷ (X) ⇒ Y Remainder ⇒ D 32 by 16 Bit ⇒ 16 Bit Divide (signed)	INH	18 14	OffffffffffffO	OffffffffffffO	----	ΔΔΔΔ
EMACS *opr16a* [2]	$(M_{(X)}:M_{(X+1)}) \times (M_{(Y)}:M_{(Y+1)}) + (M{\sim}M+3) \Rightarrow M{\sim}M+3$ 16 by 16 Bit ⇒ 32 Bit Multiply and Accumulate (signed)	Special	18 12 hh ll	ORROffffRRfWWP	ORROffffRRfWWP	----	ΔΔΔΔ
EMAXD *oprx0_xysp*	MAX((D), (M:M+1)) ⇒ D	IDX	18 1A xb	ORPf	ORfP	----	ΔΔΔΔ
EMAXD *oprx9,xysp*	MAX of 2 Unsigned 16-Bit Values	IDX1	18 1A xb ff	ORPO	ORPO		
EMAXD *oprx16,xysp*		IDX2	18 1A xb ee ff	OfRPP	OfRPP		
EMAXD [D,*xysp*]	N, Z, V and C status bits reflect result of	[D,IDX]	18 1A xb	OfIfRPf	OfIfRfP		
EMAXD [*oprx16,xysp*]	internal compare ((D) – (M:M+1))	[IDX2]	18 1A xb ee ff	OfIPRPf	OfIPRfP		
EMAXM *oprx0_xysp*	MAX((D), (M:M+1)) ⇒ M:M+1	IDX	18 1E xb	ORPW	ORPW	----	ΔΔΔΔ
EMAXM *oprx9,xysp*	MAX of 2 Unsigned 16-Bit Values	IDX1	18 1E xb ff	ORPWO	ORPWO		
EMAXM *oprx16,xysp*		IDX2	18 1E xb ee ff	OfRPWP	OfRPWP		
EMAXM [D,*xysp*]	N, Z, V and C status bits reflect result of	[D,IDX]	18 1E xb	OfIfRPW	OfIfRPW		
EMAXM [*oprx16,xysp*]	internal compare ((D) – (M:M+1))	[IDX2]	18 1E xb ee ff	OfIPRPW	OfIPRPW		
EMIND *oprx0_xysp*	MIN((D), (M:M+1)) ⇒ D	IDX	18 1B xb	ORPf	ORfP	----	ΔΔΔΔ
EMIND *oprx9,xysp*	MIN of 2 Unsigned 16-Bit Values	IDX1	18 1B xb ff	ORPO	ORPO		
EMIND *oprx16,xysp*		IDX2	18 1B xb ee ff	OfRPP	OfRPP		
EMIND [D,*xysp*]	N, Z, V and C status bits reflect result of	[D,IDX]	18 1B xb	OfIfRPf	OfIfRfP		
EMIND [*oprx16,xysp*]	internal compare ((D) – (M:M+1))	[IDX2]	18 1B xb ee ff	OfIPRPf	OfIPRfP		
EMINM *oprx0_xysp*	MIN((D), (M:M+1)) ⇒ M:M+1	IDX	18 1F xb	ORPW	ORPW	----	ΔΔΔΔ
EMINM *oprx9,xysp*	MIN of 2 Unsigned 16-Bit Values	IDX1	18 1F xb ff	ORPWO	ORPWO		
EMINM *oprx16,xysp*		IDX2	18 1F xb ee ff	OfRPWP	OfRPWP		
EMINM [D,*xysp*]	N, Z, V and C status bits reflect result of	[D,IDX]	18 1F xb	OfIfRPW	OfIfRPW		
EMINM [*oprx16,xysp*]	internal compare ((D) – (M:M+1))	[IDX2]	18 1F xb ee ff	OfIPRPW	OfIPRPW		
EMUL	(D) × (Y) ⇒ Y:D 16 by 16 Bit Multiply (unsigned)	INH	13	ffO	ffO	----	ΔΔ–Δ
EMULS	(D) × (Y) ⇒ Y:D 16 by 16 Bit Multiply (signed)	INH	18 13	OfO (if followed by page 2 instruction) OffO	OfO OfO	----	ΔΔ–Δ
EORA #*opr8i*	(A) ⊕ (M) ⇒ A	IMM	88 ii	P	P	----	ΔΔ0–
EORA *opr8a*	Exclusive-OR A with Memory	DIR	98 dd	rPf	rfP		
EORA *opr16a*		EXT	B8 hh ll	rPO	rOP		
EORA *oprx0_xysp*		IDX	A8 xb	rPf	rfP		
EORA *oprx9,xysp*		IDX1	A8 xb ff	rPO	rPO		
EORA *oprx16,xysp*		IDX2	A8 xb ee ff	frPP	frPP		
EORA [D,*xysp*]		[D,IDX]	A8 xb	fIfrPf	fIfrfP		
EORA [*oprx16,xysp*]		[IDX2]	A8 xb ee ff	fIPrPf	fIPrfP		

Notes:
1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.
2. *opr16a* is an extended address specification. Both X and Y point to source operands.

EORB #*opr8i*	(B) ⊕ (M) ⇒ B	IMM	C8 ii	P	P	----	ΔΔ0–
EORB *opr8a*	Exclusive-OR B with Memory	DIR	D8 dd	rPf	rfP		
EORB *opr16a*		EXT	F8 hh ll	rPO	rOP		
EORB *oprx0_xysp*		IDX	E8 xb	rPf	rfP		
EORB *oprx9,xysp*		IDX1	E8 xb ff	rPO	rPO		
EORB *oprx16,xysp*		IDX2	E8 xb ee ff	frPP	frPP		
EORB [D,*xysp*]		[D,IDX]	E8 xb	fIfrPf	fIfrfP		
EORB [*oprx16,xysp*]		[IDX2]	E8 xb ee ff	fIPrPf	fIPrfP		

Table A-1. Instruction Set Summary (Sheet 6 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
ETBL *oprx0_xysp*	(M:M+1)+ [(B)×((M+2:M+3) – (M:M+1))] ⇒ D 16-Bit Table Lookup and Interpolate Initialize B, and index before ETBL. <ea> points at first table entry (M:M+1) and B is fractional part of lookup value (no indirect addr. modes or extensions allowed)	IDX	18 3F xb	ORRfffffffP	ORRfffffffP	– – – –	Δ Δ – Δ ? C Bit is undefined in HC12
EXG *abcdxys,abcdxys*	(r1) ⇔ (r2) (if r1 and r2 same size) *or* $00:(r1) ⇒ r2 (if r1=8-bit; r2=16-bit) *or* ($r1_{low}$) ⇔ (r2) (if r1=16-bit; r2=8-bit) r1 and r2 may be A, B, CCR, D, X, Y, or SP	INH	B7 eb	P	P	– – – –	– – – –
FDIV	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Fractional Divide	INH	18 11	OfffffffffffO	OfffffffffffO	– – – –	– Δ Δ Δ
IBEQ *abdxys, rel9*	(cntr) + 1⇒ cntr If (cntr) = 0, then Branch else Continue to next instruction Increment Counter and Branch if = 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP	– – – –	– – – –
IBNE *abdxys, rel9*	(cntr) + 1⇒ cntr if (cntr) not = 0, then Branch; else Continue to next instruction Increment Counter and Branch if ≠ 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP	– – – –	– – – –
IDIV	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Integer Divide (unsigned)	INH	18 10	OfffffffffffO	OfffffffffffO	– – – –	– Δ 0 Δ
IDIVS	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Integer Divide (signed)	INH	18 15	OfffffffffffO	OfffffffffffO	– – – –	Δ Δ Δ Δ
INC *opr16a* INC *oprx0_xysp* INC *oprx9,xysp* INC *oprx16,xysp* INC [D,*xysp*] INC [*oprx16,xysp*] INCA INCB	(M) + $01 ⇒ M Increment Memory Byte (A) + $01 ⇒ A Increment Acc. A (B) + $01 ⇒ B Increment Acc. B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	72 hh ll 62 xb 62 xb ff 62 xb ee ff 62 xb 62 xb ee ff 42 52	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	– – – –	Δ Δ Δ –
INS	(SP) + $0001 ⇒ SP *Translates to* LEAS 1,SP	IDX	1B 81	Pf	PP[1]	– – – –	– – – –
INX	(X) + $0001 ⇒ X Increment Index Register X	INH	08	O	O	– – – –	– Δ – –
INY	(Y) + $0001 ⇒ Y Increment Index Register Y	INH	02	O	O	– – – –	– Δ – –
JMP *opr16a* JMP *oprx0_xysp* JMP *oprx9,xysp* JMP *oprx16,xysp* JMP [D,*xysp*] JMP [*oprx16,xysp*]	Routine address ⇒ PC Jump	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	06 hh ll 05 xb 05 xb ff 05 xb ee ff 05 xb 05 xb ee ff	PPP PPP PPP fPPP fIfPPP fIfPPP	PPP PPP PPP fPPP fIfPPP fIfPPP	– – – –	– – – –
Note 1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.							
JSR *opr8a* JSR *opr16a* JSR *oprx0_xysp* JSR *oprx9,xysp* JSR *oprx16,xysp* JSR [D,*xysp*] JSR [*oprx16,xysp*]	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$; Subroutine address ⇒ PC Jump to Subroutine	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	17 dd 16 hh ll 15 xb 15 xb ff 15 xb ee ff 15 xb 15 xb ee ff	SPPP SPPP PPPS PPPS fPPPS fIfPPPS fIfPPPS	PPPS PPPS PPPS PPPS fPPPS fIfPPPS fIfPPPS	– – – –	– – – –
LBCC *rel16*	Long Branch if Carry Clear (if C = 0)	REL	18 24 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	– – – –	– – – –
LBCS *rel16*	Long Branch if Carry Set (if C = 1)	REL	18 25 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	– – – –	– – – –
LBEQ *rel16*	Long Branch if Equal (if Z = 1)	REL	18 27 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	– – – –	– – – –
LBGE *rel16*	Long Branch Greater Than or Equal (if N ⊕ V = 0) (signed)	REL	18 2C qq rr	OPPP/OPO[1]	OPPP/OPO[1]	– – – –	– – – –

Table A-1. Instruction Set Summary (Sheet 7 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
LBGT *rel16*	Long Branch if Greater Than (if Z ⊕ (N ⊕ V) = 0) (signed)	REL	18 2E qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBHI *rel16*	Long Branch if Higher (if C ⊕ Z = 0) (unsigned)	REL	18 22 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBHS *rel16*	Long Branch if Higher or Same (if C = 0) (unsigned) same function as LBCC	REL	18 24 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBLE *rel16*	Long Branch if Less Than or Equal (if Z ⊕ (N ⊕ V) = 1) (signed)	REL	18 2F qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBLO *rel16*	Long Branch if Lower (if C = 1) (unsigned) *same function as LBCS*	REL	18 25 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBLS *rel16*	Long Branch if Lower or Same (if C ⊕ Z = 1) (unsigned)	REL	18 23 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBLT *rel16*	Long Branch if Less Than (if N ⊕ V = 1) (signed)	REL	18 2D qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBMI *rel16*	Long Branch if Minus (if N = 1)	REL	18 2B qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBNE *rel16*	Long Branch if Not Equal (if Z = 0)	REL	18 26 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBPL *rel16*	Long Branch if Plus (if N = 0)	REL	18 2A qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBRA *rel16*	Long Branch Always (if 1=1)	REL	18 20 qq rr	OPPP	OPPP	----	----
LBRN *rel16*	Long Branch Never (if 1 = 0)	REL	18 21 qq rr	OPO	OPO	----	----
LBVC *rel16*	Long Branch if Overflow Bit Clear (if V=0)	REL	18 28 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBVS *rel16*	Long Branch if Overflow Bit Set (if V = 1)	REL	18 29 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----

LDAA #*opr8i*
LDAA *opr8a*
LDAA *opr16a*
LDAA *oprx0_xysp*
LDAA *oprx9,xysp*
LDAA *oprx16,xysp*
LDAA [D,*xysp*]
LDAA [*oprx16,xysp*]
	(M) ⇒ A
	Load Accumulator A
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			86 ii
			96 dd
			B6 hh ll
			A6 xb
			A6 xb ff
			A6 xb ee ff
			A6 xb
			A6 xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rfP
					rOP
					rfP
					rPO
					frPP
					fIfrfP
					fIPrfP
						----	Δ Δ 0 –

LDAB #*opr8i*
LDAB *opr8a*
LDAB *opr16a*
LDAB *oprx0_xysp*
LDAB *oprx9,xysp*
LDAB *oprx16,xysp*
LDAB [D,*xysp*]
LDAB [*oprx16,xysp*]
	(M) ⇒ B
	Load Accumulator B
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			C6 ii
			D6 dd
			F6 hh ll
			E6 xb
			E6 xb ff
			E6 xb ee ff
			E6 xb
			E6 xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rfP
					rOP
					rfP
					rPO
					frPP
					fIfrfP
					fIPrfP
						----	Δ Δ 0 –

LDD #*opr16i*
LDD *opr8a*
LDD *opr16a*
LDD *oprx0_xysp*
LDD *oprx9,xysp*
LDD *oprx16,xysp*
LDD [D,*xysp*]
LDD [*oprx16,xysp*]
	(M:M+1) ⇒ A:B
	Load Double Accumulator D (A:B)
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			CC jj kk
			DC dd
			FC hh ll
			EC xb
			EC xb ff
			EC xb ee ff
			EC xb
			EC xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					OP
					RfP
					ROP
					RfP
					RPO
					fRPP
					fIfRfP
					fIPRfP
						----	Δ Δ 0 –

Note 1. OPPP/OPO indicates this instruction takes four cycles to refill the instruction queue if the branch is taken and three cycles if the branch is not taken.

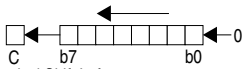
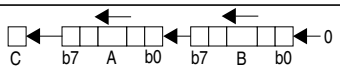
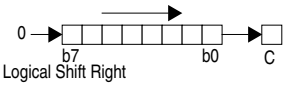
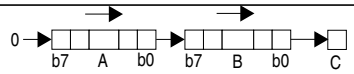
LDS #*opr16i*
LDS *opr8a*
LDS *opr16a*
LDS *oprx0_xysp*
LDS *oprx9,xysp*
LDS *oprx16,xysp*
LDS [D,*xysp*]
LDS [*oprx16,xysp*]
	(M:M+1) ⇒ SP
	Load Stack Pointer
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			CF jj kk
			DF dd
			FF hh ll
			EF xb
			EF xb ff
			EF xb ee ff
			EF xb
			EF xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					OP
					RfP
					ROP
					RfP
					RPO
					fRPP
					fIfRfP
					fIPRfP
						----	Δ Δ 0 –

LDX #*opr16i*
LDX *opr8a*
LDX *opr16a*
LDX *oprx0_xysp*
LDX *oprx9,xysp*
LDX *oprx16,xysp*
LDX [D,*xysp*]
LDX [*oprx16,xysp*]
	(M:M+1) ⇒ X
	Load Index Register X
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			CE jj kk
			DE dd
			FE hh ll
			EE xb
			EE xb ff
			EE xb ee ff
			EE xb
			EE xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					OP
					RfP
					ROP
					RfP
					RPO
					fRPP
					fIfRfP
					fIPRfP
						----	Δ Δ 0 –

Table A-1. Instruction Set Summary (Sheet 8 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C

LDY #*opr16i*
LDY *opr8a*
LDY *opr16a*
LDY *oprx0_xysp*
LDY *oprx9,xysp*
LDY *oprx16,xysp*
LDY [D,*xysp*]
LDY [*oprx16,xysp*]
	(M:M+1) ⇒ Y
	Load Index Register Y
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			CD jj kk
			DD dd
			FD hh ll
			ED xb
			ED xb ff
			ED xb ee ff
			ED xb
			ED xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					OP
					RfP
					ROP
					RfP
					RPO
					fRPP
					fIfRfP
					fIPRfP
						– – – –	Δ Δ 0 –

LEAS *oprx0_xysp* LEAS *oprx9,xysp* LEAS *oprx16,xysp*	Effective Address ⇒ SP Load Effective Address into SP	IDX IDX1 IDX2	1B xb 1B xb ff 1B xb ee ff	Pf PO PP	PP[1] PO PP	– – – –	– – – –
LEAX *oprx0_xysp* LEAX *oprx9,xysp* LEAX *oprx16,xysp*	Effective Address ⇒ X Load Effective Address into X	IDX IDX1 IDX2	1A xb 1A xb ff 1A xb ee ff	Pf PO PP	PP[1] PO PP	– – – –	– – – –
LEAY *oprx0_xysp* LEAY *oprx9,xysp* LEAY *oprx16,xysp*	Effective Address ⇒ Y Load Effective Address into Y	IDX IDX1 IDX2	19 xb 19 xb ff 19 xb ee ff	Pf PO PP	PP[1] PO PP	– – – –	– – – –
LSL *opr16a* LSL *oprx0_xysp* LSL *oprx9,xysp* LSL *oprx16,xysp* LSL [D,*xysp*] LSL [*oprx16,xysp*] LSLA LSLB	 Logical Shift Left same function as ASL Logical Shift Accumulator A to Left Logical Shift Accumulator B to Left	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	78 hh ll 68 xb 68 xb ff 68 xb ee ff 68 xb 68 xb ee ff 48 58	rPwO rPw rPwO frPPw fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	– – – –	Δ Δ Δ Δ
LSLD	 Logical Shift Left D Accumulator same function as ASLD	INH	59	O	O	– – – –	Δ Δ Δ Δ
LSR *opr16a* LSR *oprx0_xysp* LSR *oprx9,xysp* LSR *oprx16,xysp* LSR [D,*xysp*] LSR [*oprx16,xysp*] LSRA LSRB	 Logical Shift Right Logical Shift Accumulator A to Right Logical Shift Accumulator B to Right	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	74 hh ll 64 xb 64 xb ff 64 xb ee ff 64 xb 64 xb ee ff 44 54	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	– – – –	0 Δ Δ Δ
LSRD	 Logical Shift Right D Accumulator	INH	49	O	O	– – – –	0 Δ Δ Δ
MAXA *oprx0_xysp* MAXA *oprx9,xysp* MAXA *oprx16,xysp* MAXA [D,*xysp*] MAXA [*oprx16,xysp*]	MAX((A), (M)) ⇒ A MAX of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) – (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 18 xb 18 18 xb ff 18 18 xb ee ff 18 18 xb 18 18 xb ee ff	OrPf OrPO OfrPP OfIfrPf OfIPrPf	OrfP OrPO OfrPP OfIfrfP OfIPrfP	– – – –	Δ Δ Δ Δ

Note 1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.

MAXM *oprx0_xysp* MAXM *oprx9,xysp* MAXM *oprx16,xysp* MAXM [D,*xysp*] MAXM [*oprx16,xysp*]	MAX((A), (M)) ⇒ M MAX of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) – (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1C xb 18 1C xb ff 18 1C xb ee ff 18 1C xb 18 1C xb ee ff	OrPw OrPwO OfrPwP OfIfrPw OfIPrPw	OrPw OrPwO OfrPwP OfIfrPw OfIPrPw	– – – –	Δ Δ Δ Δ
MEM	μ (grade) ⇒ $M_{(Y)}$; (X) + 4 ⇒ X; (Y) + 1 ⇒ Y; A unchanged if (A) < P1 or (A) > P2 then μ = 0, else μ = MIN[((A) – P1)×S1, (P2 – (A))×S2, $FF] where: A = current crisp input value; X points at 4-byte data structure that describes a trapezoidal membership function (P1, P2, S1, S2); Y points at fuzzy input (RAM location). See *CPU12 Reference Manual* for special cases.	Special	01	RRfOw	RRfOw	– – ? –	? ? ? ?

Table A-1. Instruction Set Summary (Sheet 9 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
MINA *oprx0_xysp* MINA *oprx9,xysp* MINA *oprx16,xysp* MINA [D,*xysp*] MINA [*oprx16,xysp*]	MIN((A), (M)) ⇒ A MIN of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) – (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 19 xb 18 19 xb ff 18 19 xb ee ff 18 19 xb 18 19 xb ee ff	OrPf OrPO OfrPP OfIfrPf OfIPrPf	OrfP OrPO OfrPP OfIfrfP OfIPrfP	----	ΔΔΔΔ
MINM *oprx0_xysp* MINM *oprx9,xysp* MINM *oprx16,xysp* MINM [D,*xysp*] MINM [*oprx16,xysp*]	MIN((A), (M)) ⇒ M MIN of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) – (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1D xb 18 1D xb ff 18 1D xb ee ff 18 1D xb 18 1D xb ee ff	OrPw OrPwO OfrPwP OfIfrPw OfIPrPw	OrPw OrPwO OfrPwP OfIfrPw OfIPrPw	----	ΔΔΔΔ

MOVB #*opr8*, *opr16a*[1]
MOVB #*opr8i*, *oprx0_xysp*[1]
MOVB *opr16a*, *opr16a*[1]
MOVB *opr16a*, *oprx0_xysp*[1]
MOVB *oprx0_xysp*, *opr16a*[1]
MOVB *oprx0_xysp*, *oprx0_xysp*[1]
	$(M_1) \Rightarrow M_2$
	Memory to Memory Byte-Move (8-Bit)
		IMM-EXT
		IMM-IDX
		EXT-EXT
		EXT-IDX
		IDX-EXT
		IDX-IDX
			18 0B ii hh ll
			18 08 xb ii
			18 0C hh ll hh ll
			18 09 xb hh ll
			18 0D xb hh ll
			18 0A xb xb
				OPwP
				OPwO
				OrPwPO
				OPrPw
				OrPwP
				OrPwO
					OPwP
					OPwO
					OrPwPO
					OPrPw
					OrPwP
					OrPwO
						----	----

MOVW #*oprx16*, *opr16a*[1]
MOVW #*opr16i*, *oprx0_xysp*[1]
MOVW *opr16a*, *opr16a*[1]
MOVW *opr16a*, *oprx0_xysp*[1]
MOVW *oprx0_xysp*, *opr16a*[1]
MOVW *oprx0_xysp*, *oprx0_xysp*[1]
	$(M{:}M+1_1) \Rightarrow M{:}M+1_2$
	Memory to Memory Word-Move (16-Bit)
		IMM-EXT
		IMM-IDX
		EXT-EXT
		EXT-IDX
		IDX-EXT
		IDX-IDX
			18 03 jj kk hh ll
			18 00 xb jj kk
			18 04 hh ll hh ll
			18 01 xb hh ll
			18 05 xb hh ll
			18 02 xb xb
				OPWPO
				OPPW
				ORPWPO
				OPRPW
				ORPWP
				ORPWO
					OPWPO
					OPPW
					ORPWPO
					OPRPW
					ORPWP
					ORPWO
						----	----

MUL	(A) × (B) ⇒ A:B 8 by 8 Unsigned Multiply	INH	12	O	ffO	----	---Δ
NEG *opr16a* NEG *oprx0_xysp* NEG *oprx9,xysp* NEG *oprx16,xysp* NEG [D,*xysp*] NEG [*oprx16,xysp*] NEGA NEGB	0 – (M) ⇒ M *equivalent to* $(\overline{M}) + 1 \Rightarrow M$ Two's Complement Negate 0 – (A) ⇒ A *equivalent to* $(\overline{A}) + 1 \Rightarrow A$ Negate Accumulator A 0 – (B) ⇒ B *equivalent to* $(\overline{B}) + 1 \Rightarrow B$ Negate Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	70 hh ll 60 xb 60 xb ff 60 xb ee ff 60 xb 60 xb ee ff 40 50	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	ΔΔΔΔ
NOP	No Operation	INH	A7	O	O	----	----
ORAA #*opr8i* ORAA *opr8a* ORAA *opr16a* ORAA *oprx0_xysp* ORAA *oprx9,xysp* ORAA *oprx16,xysp* ORAA [D,*xysp*] ORAA [*oprx16,xysp*]	(A) + (M) ⇒ A Logical OR A with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8A ii 9A dd BA hh ll AA xb AA xb ff AA xb ee ff AA xb AA xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP rPO frPP fIfrfP fIPrfP	----	ΔΔ0–

Note 1. The first operand in the source code statement specifies the source for the move.

ORAB #*opr8i* ORAB *opr8a* ORAB *opr16a* ORAB *oprx0_xysp* ORAB *oprx9,xysp* ORAB *oprx16,xysp* ORAB [D,*xysp*] ORAB [*oprx16,xysp*]	(B) + (M) ⇒ B Logical OR B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CA ii DA dd FA hh ll EA xb EA xb ff EA xb ee ff EA xb EA xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP rPO frPP fIfrfP fIPrfP	----	ΔΔ0–
ORCC #*opr8i*	(CCR) + M ⇒ CCR Logical OR CCR with Memory	IMM	14 ii	P	P	⇑–⇑⇑	⇑⇑⇑⇑
PSHA	(SP) – 1 ⇒ SP; (A) ⇒ $M_{(SP)}$ Push Accumulator A onto Stack	INH	36	Os	Os	----	----
PSHB	(SP) – 1 ⇒ SP; (B) ⇒ $M_{(SP)}$ Push Accumulator B onto Stack	INH	37	Os	Os	----	----
PSHC	(SP) – 1 ⇒ SP; (CCR) ⇒ $M_{(SP)}$ Push CCR onto Stack	INH	39	Os	Os	----	----
PSHD	(SP) – 2 ⇒ SP; (A:B) ⇒ $M_{(SP)}{:}M_{(SP+1)}$ Push D Accumulator onto Stack	INH	3B	OS	OS	----	----
PSHX	(SP) – 2 ⇒ SP; $(X_H{:}X_L) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$ Push Index Register X onto Stack	INH	34	OS	OS	----	----
PSHY	(SP) – 2 ⇒ SP; $(Y_H{:}Y_L) \Rightarrow M_{(SP)}{:}M_{(SP+1)}$ Push Index Register Y onto Stack	INH	35	OS	OS	----	----

Table A-1. Instruction Set Summary (Sheet 10 of 14)

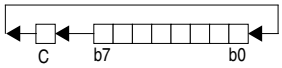
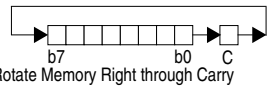
Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
PULA	$(M_{(SP)}) \Rightarrow A$; $(SP) + 1 \Rightarrow SP$ Pull Accumulator A from Stack	INH	32	ufO	ufO	----	----
PULB	$(M_{(SP)}) \Rightarrow B$; $(SP) + 1 \Rightarrow SP$ Pull Accumulator B from Stack	INH	33	ufO	ufO	----	----
PULC	$(M_{(SP)}) \Rightarrow CCR$; $(SP) + 1 \Rightarrow SP$ Pull CCR from Stack	INH	38	ufO	ufO	Δ⇓ΔΔ	ΔΔΔΔ
PULD	$(M_{(SP)}:M_{(SP+1)}) \Rightarrow A:B$; $(SP) + 2 \Rightarrow SP$ Pull D from Stack	INH	3A	UfO	UfO	----	----
PULX	$(M_{(SP)}:M_{(SP+1)}) \Rightarrow X_H:X_L$; $(SP) + 2 \Rightarrow SP$ Pull Index Register X from Stack	INH	30	UfO	UfO	----	----
PULY	$(M_{(SP)}:M_{(SP+1)}) \Rightarrow Y_H:Y_L$; $(SP) + 2 \Rightarrow SP$ Pull Index Register Y from Stack	INH	31	UfO	UfO	----	----
REV	MIN-MAX rule evaluation Find smallest rule input (MIN). Store to rule outputs unless fuzzy output is already larger (MAX). For rule weights see REVW. Each rule input is an 8-bit offset from the base address in Y. Each rule output is an 8-bit offset from the base address in Y. $FE separates rule inputs from rule outputs. $FF terminates the rule list. REV may be interrupted.	Special	18 3A	Orf(t,tx)O (exit + re-entry replaces comma above if interrupted) ff + Orf(t,	Orf(t,tx)O ff + Orf(t,	--?-	??Δ?
REVW	MIN-MAX rule evaluation Find smallest rule input (MIN), Store to rule outputs unless fuzzy output is already larger (MAX). Rule weights supported, optional. Each rule input is the 16-bit address of a fuzzy input. Each rule output is the 16-bit address of a fuzzy output. The value $FFFE separates rule inputs from rule outputs. $FFFF terminates the rule list. REVW may be interrupted.	Special	18 3B	ORf(t,Tx)O (loop to read weight if enabled) (r,RfRf) (exit + re-entry replaces comma above if interrupted) ffff + ORf(t,	ORf(t,Tx)O (r,RfRf) fff + ORf(t,	--?-	??Δ!
ROL *opr16a* ROL *oprx0_xysp* ROL *oprx9,xysp* ROL *oprx16,xysp* ROL [D,*xysp*] ROL [*oprx16,xysp*] ROLA ROLB	 Rotate Memory Left through Carry Rotate A Left through Carry Rotate B Left through Carry	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	75 hh ll 65 xb 65 xb ff 65 xb ee ff 65 xb 65 xb ee ff 45 55	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	ΔΔΔΔ
ROR *opr16a* ROR *oprx0_xysp* ROR *oprx9,xysp* ROR *oprx16,xysp* ROR [D,*xysp*] ROR [*oprx16,xysp*] RORA RORB	 Rotate Memory Right through Carry Rotate A Right through Carry Rotate B Right through Carry	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	76 hh ll 66 xb 66 xb ff 66 xb ee ff 66 xb 66 xb ee ff 46 56	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	ΔΔΔΔ
RTC	$(M_{(SP)}) \Rightarrow PPAGE$; $(SP) + 1 \Rightarrow SP$; $(M_{(SP)}:M_{(SP+1)}) \Rightarrow PC_H:PC_L$; $(SP) + 2 \Rightarrow SP$ Return from Call	INH	0A	uUnfPPP	uUnPPP	----	----
RTI	$(M_{(SP)}) \Rightarrow CCR$; $(SP) + 1 \Rightarrow SP$ $(M_{(SP)}:M_{(SP+1)}) \Rightarrow B:A$; $(SP) + 2 \Rightarrow SP$ $(M_{(SP)}:M_{(SP+1)}) \Rightarrow X_H:X_L$; $(SP) + 4 \Rightarrow SP$ $(M_{(SP)}:M_{(SP+1)}) \Rightarrow PC_H:PC_L$; $(SP) - 2 \Rightarrow SP$ $(M_{(SP)}:M_{(SP+1)}) \Rightarrow Y_H:Y_L$; $(SP) + 4 \Rightarrow SP$ Return from Interrupt	INH	0B	uUUUUPPP (with interrupt pending) uUUUUVfPPP	uUUUUPPP uUUUUfVfPPP	Δ⇓ΔΔ	ΔΔΔΔ
RTS	$(M_{(SP)}:M_{(SP+1)}) \Rightarrow PC_H:PC_L$; $(SP) + 2 \Rightarrow SP$ Return from Subroutine	INH	3D	UfPPP	UfPPP	----	----

Table A-1. Instruction Set Summary (Sheet 11 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
SBA	(A) – (B) ⇒ A Subtract B from A	INH	18 16	OO	OO	– – – –	Δ Δ Δ Δ
SBCA #*opr8i* SBCA *opr8a* SBCA *opr16a* SBCA *oprx0_xysp* SBCA *oprx9,xysp* SBCA *oprx16,xysp* SBCA [D,*xysp*] SBCA [*oprx16,xysp*]	(A) – (M) – C ⇒ A Subtract with Borrow from A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	82 ii 92 dd B2 hh ll A2 xb A2 xb ff A2 xb ee ff A2 xb A2 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP rPO frPP fIfrfP fIPrfP	– – – –	Δ Δ Δ Δ
SBCB #*opr8i* SBCB *opr8a* SBCB *opr16a* SBCB *oprx0_xysp* SBCB *oprx9,xysp* SBCB *oprx16,xysp* SBCB [D,*xysp*] SBCB [*oprx16,xysp*]	(B) – (M) – C ⇒ B Subtract with Borrow from B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C2 ii D2 dd F2 hh ll E2 xb E2 xb ff E2 xb ee ff E2 xb E2 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP rPO frPP fIfrfP fIPrfP	– – – –	Δ Δ Δ Δ
SEC	1 ⇒ C *Translates to* ORCC #$01	IMM	14 01	P	P	– – – –	– – – 1
SEI	1 ⇒ I; (inhibit I interrupts) *Translates to* ORCC #$10	IMM	14 10	P	P	– – – 1	– – – –
SEV	1 ⇒ V *Translates to* ORCC #$02	IMM	14 02	P	P	– – – –	– – 1 –
SEX *abc,dxys*	$00:(r1) ⇒ r2 if r1, bit 7 is 0 *or* $FF:(r1) ⇒ r2 if r1, bit 7 is 1 Sign Extend 8-bit r1 to 16-bit r2 r1 may be A, B, or CCR r2 may be D, X, Y, or SP *Alternate mnemonic for* TFR r1, r2	INH	B7 eb	P	P	– – – –	– – – –
STAA *opr8a* STAA *opr16a* STAA *oprx0_xysp* STAA *oprx9,xysp* STAA *oprx16,xysp* STAA [D,*xysp*] STAA [*oprx16,xysp*]	(A) ⇒ M Store Accumulator A to Memory	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5A dd 7A hh ll 6A xb 6A xb ff 6A xb ee ff 6A xb 6A xb ee ff	Pw PwO Pw PwO PwP PIfw PIPw	Pw wOP Pw PwO PwP PIfPw PIPPw	– – – –	Δ Δ 0 –
STAB *opr8a* STAB *opr16a* STAB *oprx0_xysp* STAB *oprx9,xysp* STAB *oprx16,xysp* STAB [D,*xysp*] STAB [*oprx16,xysp*]	(B) ⇒ M Store Accumulator B to Memory	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5B dd 7B hh ll 6B xb 6B xb ff 6B xb ee ff 6B xb 6B xb ee ff	Pw PwO Pw PwO PwP PIfw PIPw	Pw wOP Pw PwO PwP PIfPw PIPPw	– – – –	Δ Δ 0 –
STD *opr8a* STD *opr16a* STD *oprx0_xysp* STD *oprx9,xysp* STD *oprx16,xysp* STD [D,*xysp*] STD [*oprx16,xysp*]	(A) ⇒ M, (B) ⇒ M+1 Store Double Accumulator	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5C dd 7C hh ll 6C xb 6C xb ff 6C xb ee ff 6C xb 6C xb ee ff	PW PWO PW PWO PWP PIfW PIPW	PW WOP PW PWO PWP PIfPW PIPPW	– – – –	Δ Δ 0 –
STOP	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (Y_H:Y_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (X_H:X_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (B:A) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 1 ⇒ SP; (CCR) ⇒ $M_{(SP)}$; STOP All Clocks Registers stacked to allow quicker recovery by interrupt. If S control bit = 1, the STOP instruction is disabled and acts like a two-cycle NOP.	INH	18 3E	(entering STOP) OOSSSSsf (exiting STOP) fVfPPP (continue) ff (if STOP disabled) OO	 OOSSSfSs fVfPPP fO OO	– – – –	– – – –

Table A-1. Instruction Set Summary (Sheet 12 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
STS *opr8a* STS *opr16a* STS *oprx0_xysp* STS *oprx9,xysp* STS *oprx16,xysp* STS [D,*xysp*] STS [*oprx16,xysp*]	$(SP_H:SP_L) \Rightarrow M:M+1$ Store Stack Pointer	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5F dd 7F hh ll 6F xb 6F xb ff 6F xb ee ff 6F xb 6F xb ee ff	PW PWO PW PWO PWP PIfW PIPW	PW WOP PW PWO PWP PIfPW PIPPW	----	Δ Δ 0 –
STX *opr8a* STX *opr16a* STX *oprx0_xysp* STX *oprx9,xysp* STX *oprx16,xysp* STX [D,*xysp*] STX [*oprx16,xysp*]	$(X_H:X_L) \Rightarrow M:M+1$ Store Index Register X	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5E dd 7E hh ll 6E xb 6E xb ff 6E xb ee ff 6E xb 6E xb ee ff	PW PWO PW PWO PWP PIfW PIPW	PW WOP PW PWO PWP PIfPW PIPPW	----	Δ Δ 0 –
STY *opr8a* STY *opr16a* STY *oprx0_xysp* STY *oprx9,xysp* STY *oprx16,xysp* STY [D,*xysp*] STY [*oprx16,xysp*]	$(Y_H:Y_L) \Rightarrow M:M+1$ Store Index Register Y	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5D dd 7D hh ll 6D xb 6D xb ff 6D xb ee ff 6D xb 6D xb ee ff	PW PWO PW PWO PWP PIfW PIPW	PW WOP PW PWO PWP PIfPW PIPPW	----	Δ Δ 0 –

SUBA #*opr8i*
SUBA *opr8a*
SUBA *opr16a*
SUBA *oprx0_xysp*
SUBA *oprx9,xysp*
SUBA *oprx16,xysp*
SUBA [D,*xysp*]
SUBA [*oprx16,xysp*]
	$(A) - (M) \Rightarrow A$
	Subtract Memory from Accumulator A
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			80 ii
			90 dd
			B0 hh ll
			A0 xb
			A0 xb ff
			A0 xb ee ff
			A0 xb
			A0 xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rfP
					rOP
					rfP
					rPO
					frPP
					fIfrfP
					fIPrfP
						----	Δ Δ Δ Δ

SUBB #*opr8i*
SUBB *opr8a*
SUBB *opr16a*
SUBB *oprx0_xysp*
SUBB *oprx9,xysp*
SUBB *oprx16,xysp*
SUBB [D,*xysp*]
SUBB [*oprx16,xysp*]
	$(B) - (M) \Rightarrow B$
	Subtract Memory from Accumulator B
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			C0 ii
			D0 dd
			F0 hh ll
			E0 xb
			E0 xb ff
			E0 xb ee ff
			E0 xb
			E0 xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rfP
					rOP
					rfP
					rPO
					frPP
					fIfrfP
					fIPrfP
						----	Δ Δ Δ Δ

SUBD #*opr16i*
SUBD *opr8a*
SUBD *opr16a*
SUBD *oprx0_xysp*
SUBD *oprx9,xysp*
SUBD *oprx16,xysp*
SUBD [D,*xysp*]
SUBD [*oprx16,xysp*]
	$(D) - (M:M+1) \Rightarrow D$
	Subtract Memory from D (A:B)
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			83 jj kk
			93 dd
			B3 hh ll
			A3 xb
			A3 xb ff
			A3 xb ee ff
			A3 xb
			A3 xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					OP
					RfP
					ROP
					RfP
					RPO
					fRPP
					fIfRfP
					fIPRfP
						----	Δ Δ Δ Δ

SWI	(SP) – 2 ⇒ SP; $RTN_H:RTN_L \Rightarrow M_{(SP)}:M_{(SP+1)}$; (SP) – 2 ⇒ SP; $(Y_H:Y_L) \Rightarrow M_{(SP)}:M_{(SP+1)}$; (SP) – 2 ⇒ SP; $(X_H:X_L) \Rightarrow M_{(SP)}:M_{(SP+1)}$; (SP) – 2 ⇒ SP; (B:A) ⇒ $M_{(SP)}:M_{(SP+1)}$; (SP) – 1 ⇒ SP; (CCR) ⇒ $M_{(SP)}$ 1 ⇒ I; (SWI Vector) ⇒ PC Software Interrupt	INH	3F	VSPSSPSsP*	VSPSSPSsP*	---1	----
				(for Reset) VfPPP	VfPPP	1 1 – 1	----
*The CPU also uses the SWI microcode sequence for hardware interrupts and unimplemented opcode traps. Reset uses the VfPPP variation of this sequence.							
TAB	(A) ⇒ B Transfer A to B	INH	18 0E	OO	OO	----	Δ Δ 0 –
TAP	(A) ⇒ CCR *Translates to* TFR A , CCR	INH	B7 02	P	P	Δ ⇓ Δ Δ	Δ Δ Δ Δ
TBA	(B) ⇒ A Transfer B to A	INH	18 0F	OO	OO	----	Δ Δ 0 –
TBEQ *abdxys,rel9*	If (cntr) = 0, then Branch; else Continue to next instruction Test Counter and Branch if Zero (cntr = A, B, D, X,Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP	----	----

Table A-1. Instruction Set Summary (Sheet 13 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	Access Detail M68HC12	S X H I	N Z V C
TBL *oprx0_xysp*	(M) + [(B) × ((M+1) – (M))] ⇒ A 8-Bit Table Lookup and Interpolate Initialize B, and index before TBL. <ea> points at first 8-bit table entry (M) and B is fractional part of lookup value. (no indirect addressing modes or extensions allowed)	IDX	18 3D xb	ORfffP	OrrffffP	----	Δ Δ – Δ ? C Bit is undefined in HC12
TBNE *abdxys,rel9*	If (cntr) not = 0, then Branch; else Continue to next instruction Test Counter and Branch if Not Zero (cntr = A, B, D, X,Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP	----	----
TFR *abcdxys,abcdxys*	(r1) ⇒ r2 *or* $00:(r1) ⇒ r2 *or* (r1[7:0]) ⇒ r2 Transfer Register to Register r1 and r2 may be A, B, CCR, D, X, Y, or SP	INH	B7 eb	P	P	---- or Δ ⇓ Δ Δ	---- Δ Δ Δ Δ
TPA	(CCR) ⇒ A *Translates to* TFR CCR ,A	INH	B7 20	P	P	----	----
TRAP *trapnum*	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (Y_H:Y_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (X_H:X_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (B:A) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 1 ⇒ SP; (CCR) ⇒ $M_{(SP)}$ 1 ⇒ I; (TRAP Vector) ⇒ PC Unimplemented opcode trap	INH	18 tn tn = $30–$39 or $40–$FF	OVSPSSPSsP	OfVSPSSPSsP	---1	----
TST *opr16a* TST *oprx0_xysp* TST *oprx9,xysp* TST *oprx16,xysp* TST [D,*xysp*] TST [*oprx16,xysp*] TSTA TSTB	(M) – 0 Test Memory for Zero or Minus (A) – 0 Test A for Zero or Minus (B) – 0 Test B for Zero or Minus	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	F7 hh ll E7 xb E7 xb ff E7 xb ee ff E7 xb E7 xb ee ff 97 D7	rPO rPf rPO frPP fIfrPf fIPrPf O O	rOP rfP rPO frPP fIfrfP fIPrfP O O	----	Δ Δ 0 0
TSX	(SP) ⇒ X *Translates to* TFR SP,X	INH	B7 75	P	P	----	----
TSY	(SP) ⇒ Y *Translates to* TFR SP,Y	INH	B7 76	P	P	----	----
TXS	(X) ⇒ SP *Translates to* TFR X,SP	INH	B7 57	P	P	----	----
TYS	(Y) ⇒ SP *Translates to* TFR Y,SP	INH	B7 67	P	P	----	----
WAI	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (Y_H:Y_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (X_H:X_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (B:A) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 1 ⇒ SP; (CCR) ⇒ $M_{(SP)}$; WAIT for interrupt	INH	3E	OSSSSsf (after interrupt) fVfPPP	OSSSfSsf VfPPP	---- or ---1 or –1–1	---- or ---- ----

Table A-1. Instruction Set Summary (Sheet 14 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12 / M68HC12	S X H I	N Z V C
WAV	$\sum_{i=1}^{B} S_i F_i \Rightarrow Y{:}D$ and $\sum_{i=1}^{B} F_i \Rightarrow X$ Calculate Sum of Products and Sum of Weights for Weighted Average Calculation Initialize B, X, and Y before WAV. B specifies number of elements. X points at first element in S_i list. Y points at first element in F_i list. All S_i and F_i elements are 8-bits. If interrupted, six extra bytes of stack used for intermediate values	Special	18 3C	Of(frr,ffff)O Off(frr,fffff)O (add if interrupt) SSS + UUUrr, SSSf + UUUrr	--?-	?Δ??
wavr pseudo-instruction	*see* WAV Resume executing an interrupted WAV instruction (recover intermediate results from stack rather than initializing them to zero)	Special	3C	UUUrr,ffff UUUrrfffff (frr,ffff)O (frr,fffff)O (exit + re-entry replaces comma above if interrupted) SSS + UUUrr, SSSf + UUUrr	--?-	?Δ??
XGDX	(D) ⇔ (X) *Translates to* EXG D, X	INH	B7 C5	P P	----	----
XGDY	(D) ⇔ (Y) *Translates to* EXG D, Y	INH	B7 C6	P P	----	----