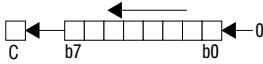
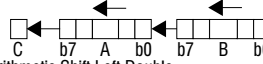


Table A-1. Instruction Set Summary (Sheet 1 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
ABA	$(A) + (B) \Rightarrow A$ Add Accumulators A and B	INH	18 06	00	00	--Δ-	ΔΔΔΔ
ABX	$(B) + (X) \Rightarrow X$ <i>Translates to LEAX B,X</i>	IDX	1A E5	Pf	PP ¹	----	----
ABY	$(B) + (Y) \Rightarrow Y$ <i>Translates to LEAY B,Y</i>	IDX	19 ED	Pf	PP ¹	----	----
ADCA #opr8i ADCA opr8a ADCA opr16a ADCA oprx0_xysp ADCA oprx9_xysp ADCA oprx16_xysp ADCA [D,xysp] ADCA [opr16,xysp]	$(A) + (M) + C \Rightarrow A$ Add with Carry to A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	89 ii 99 dd B9 hh 11 A9 xb A9 xb ff A9 xb ee ff A9 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rOP rPf rPO frPP fIfrPf fIPrPf	--Δ-	ΔΔΔΔ
ADCB #opr8i ADCB opr8a ADCB opr16a ADCB oprx0_xysp ADCB oprx9_xysp ADCB oprx16_xysp ADCB [D,xysp] ADCB [opr16,xysp]	$(B) + (M) + C \Rightarrow B$ Add with Carry to B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C9 ii D9 dd F9 hh 11 E9 xb E9 xb ff E9 xb ee ff E9 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rOP rPf rPO frPP fIfrPf fIPrPf	--Δ-	ΔΔΔΔ
ADDA #opr8i ADDA opr8a ADDA opr16a ADDA oprx0_xysp ADDA oprx9_xysp ADDA oprx16_xysp ADDA [D,xysp] ADDA [opr16,xysp]	$(A) + (M) \Rightarrow A$ Add without Carry to A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8B ii 9B dd BB hh 11 AB xb AB xb ff AB xb ee ff AB xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rOP rPf rPO frPP fIfrPf fIPrPf	--Δ-	ΔΔΔΔ
ADDB #opr8i ADDB opr8a ADDB opr16a ADDB oprx0_xysp ADDB oprx9_xysp ADDB oprx16_xysp ADDB [D,xysp] ADDB [opr16,xysp]	$(B) + (M) \Rightarrow B$ Add without Carry to B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CB ii DB dd FB hh 11 EB xb EB xb ff EB xb ee ff EB xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rOP rPf rPO frPP fIfrPf fIPrPf	--Δ-	ΔΔΔΔ
ADDD #opr16i ADDD opr8a ADDD opr16a ADDD oprx0_xysp ADDD oprx9_xysp ADDD oprx16_xysp ADDD [D,xysp] ADDD [opr16,xysp]	$(A:B) + (M:M+1) \Rightarrow A:B$ Add 16-Bit to D (A:B)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C3 jj kk D3 dd F3 hh 11 E3 xb E3 xb ff E3 xb ee ff E3 xb ee ff	PO RPf RPO RPf RPO FRPP fIfrPf fIPrPf	OP RfP ROP RfP RPO FRPP fIfrPf fIPrPf	----	ΔΔΔΔ
ANDA #opr8i ANDA opr8a ANDA opr16a ANDA oprx0_xysp ANDA oprx9_xysp ANDA oprx16_xysp ANDA [D,xysp] ANDA [opr16,xysp]	$(A) \bullet (M) \Rightarrow A$ Logical AND A with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	84 ii 94 dd B4 hh 11 A4 xb A4 xb ff A4 xb ee ff A4 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rOP rPf rPO frPP fIfrPf fIPrPf	----	ΔΔ0-
ANDB #opr8i ANDB opr8a ANDB opr16a ANDB oprx0_xysp ANDB oprx9_xysp ANDB oprx16_xysp ANDB [D,xysp] ANDB [opr16,xysp]	$(B) \bullet (M) \Rightarrow B$ Logical AND B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C4 ii D4 dd F4 hh 11 E4 xb E4 xb ff E4 xb ee ff E4 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rOP rPf rPO frPP fIfrPf fIPrPf	----	ΔΔ0-
ANDCC #opr8i	$(CCR) \bullet (M) \Rightarrow CCR$ 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		S X H I	N Z V C
				HCS12	M68HC12		
ASL <i>opr16a</i> ASL <i>opr0_xysp</i> ASL <i>opr9_xysp</i> ASL <i>opr16_xysp</i> ASL [D, <i>xysp</i>] ASL [<i>opr16_xysp</i>] 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 fIfPw fIPrPw O O	rOPw rPw rPOw frPPw fIfPw fIPrPw O O	----	Δ Δ Δ Δ
ASLD	 Arithmetic Shift Left Double	INH	59	O	O	----	Δ Δ Δ Δ
ASR <i>opr16a</i> ASR <i>opr0_xysp</i> ASR <i>opr9_xysp</i> ASR <i>opr16_xysp</i> ASR [D, <i>xysp</i>] ASR [<i>opr16_xysp</i>] 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 fIfPw fIPrPw O O	rOPw rPw rPOw frPPw fIfPw fIPrPw O O	----	Δ Δ Δ Δ
BCC <i>rel8</i>	Branch if Carry Clear (if C = 0)	REL	24 rr	PPP/P ¹	PPP/P ¹	----	----
BCLR <i>opr8a, msk8</i> BCLR <i>opr16a, msk8</i> BCLR <i>opr0_xysp, msk8</i> BCLR <i>opr9_xysp, msk8</i> BCLR <i>opr16_xysp, msk8</i>	(M) • (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	----	Δ Δ 0 -
BCS <i>rel8</i>	Branch if Carry Set (if C = 1)	REL	25 rr	PPP/P ¹	PPP/P ¹	----	----
BEQ <i>rel8</i>	Branch if Equal (if Z = 1)	REL	27 rr	PPP/P ¹	PPP/P ¹	----	----
BGE <i>rel8</i>	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 <i>rel8</i>	Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	2E rr	PPP/P ¹	PPP/P ¹	----	----
BHI <i>rel8</i>	Branch if Higher (if C + Z = 0) (unsigned)	REL	22 rr	PPP/P ¹	PPP/P ¹	----	----
BHS <i>rel8</i>	Branch if Higher or Same (if C = 0) (unsigned) same function as BCC	REL	24 rr	PPP/P ¹	PPP/P ¹	----	----
BITA # <i>opr8i</i> BITA <i>opr8a</i> BITA <i>opr16a</i> BITA <i>opr0_xysp</i> BITA <i>opr9_xysp</i> BITA <i>opr16_xysp</i> BITA [D, <i>xysp</i>] BITA [<i>opr16_xysp</i>]	(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 fIfPw fIPrPf	P rPf rOP rPf rPO frPP fIfPw fIPrPf	----	Δ Δ 0 -
BITB # <i>opr8i</i> BITB <i>opr8a</i> BITB <i>opr16a</i> BITB <i>opr0_xysp</i> BITB <i>opr9_xysp</i> BITB <i>opr16_xysp</i> BITB [D, <i>xysp</i>] BITB [<i>opr16_xysp</i>]	(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 fIfPw fIPrPf	P rPf rOP rPf rPO frPP fIfPw fIPrPf	----	Δ Δ 0 -
BLE <i>rel8</i>	Branch if Less Than or Equal (if Z + (N ⊕ V) = 1) (signed)	REL	2F rr	PPP/P ¹	PPP/P ¹	----	----
BLO <i>rel8</i>	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		S X H I	N Z V C
				HCS12	M68HC12		
BLS <i>rel8</i>	Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	23 rr	PPP/P ¹	PPP/P ¹	----	----
BLT <i>rel8</i>	Branch if Less Than (if N ⊕ V = 1) (signed)	REL	2D rr	PPP/P ¹	PPP/P ¹	----	----
BMI <i>rel8</i>	Branch if Minus (if N = 1)	REL	2B rr	PPP/P ¹	PPP/P ¹	----	----
BNE <i>rel8</i>	Branch if Not Equal (if Z = 0)	REL	26 rr	PPP/P ¹	PPP/P ¹	----	----
BPL <i>rel8</i>	Branch if Plus (if N = 0)	REL	2A rr	PPP/P ¹	PPP/P ¹	----	----
BRA <i>rel8</i>	Branch Always (if 1 = 1)	REL	20 rr	PPP	PPP	----	----
BRCLR <i>opr8a, msk8, rel8</i> BRCLR <i>opr16a, msk8, rel8</i> BRCLR <i>opr0_xysp, msk8, rel8</i> BRCLR <i>opr9_xysp, msk8, rel8</i> BRCLR <i>opr16_xysp, msk8, rel8</i>	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 rfPPP frPfPPP	----	----
BRN <i>rel8</i>	Branch Never (if 1 = 0)	REL	21 rr	P	P	----	----
BRSET <i>opr8, msk8, rel8</i> BRSET <i>opr16a, msk8, rel8</i> BRSET <i>opr0_xysp, msk8, rel8</i> BRSET <i>opr9_xysp, msk8, rel8</i> BRSET <i>opr16_xysp, msk8, rel8</i>	Branch if (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 rfPPP frPfPPP	----	----
BSET <i>opr8, msk8</i> BSET <i>opr16a, msk8</i> BSET <i>opr0_xysp, msk8</i> BSET <i>opr9_xysp, msk8</i> BSET <i>opr16_xysp, msk8</i>	(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	rPwO rPwP rPwO rPwP frPwOP	----	Δ Δ 0 -
BSR <i>rel8</i>	(SP) - 2 ⇒ SP; RTN _H :RTN _L ⇒ M _(SP) :M _(SP+1) Subroutine address ⇒ PC Branch to Subroutine	REL	07 rr	SPPP	PPPS	----	----
BVC <i>rel8</i>	Branch if Overflow Bit Clear (if V = 0)	REL	28 rr	PPP/P ¹	PPP/P ¹	----	----
BVS <i>rel8</i>	Branch if Overflow Bit Set (if V = 1)	REL	29 rr	PPP/P ¹	PPP/P ¹	----	----
CALL <i>opr16a, page</i> CALL <i>opr0_xysp, page</i> CALL <i>opr9_xysp, page</i> CALL <i>opr16_xysp, page</i> CALL [D, <i>xysp</i>] CALL [<i>opr16, xysp</i>]	(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 <i>opr16a</i> CLR <i>opr0_xysp</i> CLR <i>opr9_xysp</i> CLR <i>opr16_xysp</i> CLR [D, <i>xysp</i>] CLR [<i>opr16, xysp</i>] 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	----	0100
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.

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
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 11 A1 xb A1 xb ff A1 xb ee ff A1 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rOP rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ Δ Δ
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 11 E1 xb E1 xb ff E1 xb ee ff E1 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rOP rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ Δ Δ
COM opr16a COM oprx0_xysp COM oprx9_xysp COM oprx16_xysp COM [D,xysp] COM [oprx16,xysp] COMA COMB	(M) ⇒ M equivalent to \$FF – (M) ⇒ M 1's Complement Memory Location (A) ⇒ A Complement Accumulator A (B) ⇒ B Complement Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	71 hh 11 61 xb 61 xb ff 61 xb ee ff 61 xb ee ff 41 51	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPwO 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 11 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 fIfrPf fIPrPf	----	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 11 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 fIfrPf fIPrPf	----	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 11 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 fIfrPf fIPrPf	----	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 11 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 fIfrPf fIPrPf	----	Δ Δ Δ Δ
DAA	Adjust Sum to BCD Decimal Adjust Accumulator A	INH	18 07	Ofo	Ofo	----	Δ Δ ? Δ
DBEQ abdxys, rel9	(cnt) – 1 ⇒ cnt if (cnt) = 0, then Branch else Continue to next instruction Decrement Counter and Branch if = 0 (cnt = A, B, D, X, Y, or SP)	REL (9-bit)	04 1b 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		S X H I	N Z V C
				HCS12	M68HC12		
DBNE <i>abdxys, rel9</i>	(cnt) – 1 ⇒ cnt If (cnt) not = 0, then Branch; else Continue to next instruction Decrement Counter and Branch if ≠ 0 (cnt = A, B, D, X, Y, or SP)	REL (9-bit)	04 1b rr	PPP (branch) PPO (no branch)	PPP	----	----
DEC <i>opr16a</i> DEC <i>opr0_xysp</i> DEC <i>opr9_xysp</i> DEC <i>opr16_xysp</i> DEC [D,xysp] DEC [<i>opr16_xysp</i>] DECA DECB	(M) – \$01 ⇒ M Decrement Memory Location (A) – \$01 ⇒ A Decrement A (B) – \$01 ⇒ B Decrement B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	73 hh 1l 63 xb 63 xb ff 63 xb ee ff 63 xb ee ff 43 53	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	Δ Δ Δ –
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	ffffffffff0	ffffffffff0	----	Δ Δ Δ Δ
EDIVS	(Y:D) ÷ (X) ⇒ Y Remainder ⇒ D 32 by 16 Bit ⇒ 16 Bit Divide (signed)	INH	18 14	0ffffffffff0	0ffffffffff0	----	Δ Δ Δ Δ
EMACS <i>opr16a</i> ²	(M _(X) :M _(X+1)) × (M _(Y) :M _(Y+1)) + (M~M+3) ⇒ M~M+3 16 by 16 Bit ⇒ 32 Bit Multiply and Accumulate (signed)	Special	18 12 hh 1l	ORROffFRfWWP	ORROffFRfWWP	----	Δ Δ Δ Δ
EMAXD <i>opr0_xysp</i> EMAXD <i>opr9_xysp</i> EMAXD <i>opr16_xysp</i> EMAXD [D,xysp] EMAXD [<i>opr16_xysp</i>]	MAX((D), (M:M+1)) ⇒ D MAX of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1A xb 18 1A xb ff 18 1A xb ee ff 18 1A xb 18 1A xb ee ff	ORPf ORPO OfRPP OfIfrPf OfIPrPf	ORfP ORPO OfRPP OfIfrfP OfIPrfP	----	Δ Δ Δ Δ
EMAXM <i>opr0_xysp</i> EMAXM <i>opr9_xysp</i> EMAXM <i>opr16_xysp</i> EMAXM [D,xysp] EMAXM [<i>opr16_xysp</i>]	MAX((D), (M:M+1)) ⇒ M:M+1 MAX of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1E xb 18 1E xb ff 18 1E xb ee ff 18 1E xb 18 1E xb ee ff	ORPW ORPWO OfRPWP OfIfrPW OfIPrPW	ORPW ORPWO OfRPWP OfIfrPW OfIPrPW	----	Δ Δ Δ Δ
EMIND <i>opr0_xysp</i> EMIND <i>opr9_xysp</i> EMIND <i>opr16_xysp</i> EMIND [D,xysp] EMIND [<i>opr16_xysp</i>]	MIN((D), (M:M+1)) ⇒ D MIN of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1B xb 18 1B xb ff 18 1B xb ee ff 18 1B xb 18 1B xb ee ff	ORPf ORPO OfRPP OfIfrPf OfIPrPf	ORfP ORPO OfRPP OfIfrfP OfIPrfP	----	Δ Δ Δ Δ
EMINM <i>opr0_xysp</i> EMINM <i>opr9_xysp</i> EMINM <i>opr16_xysp</i> EMINM [D,xysp] EMINM [<i>opr16_xysp</i>]	MIN((D), (M:M+1)) ⇒ M:M+1 MIN of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1F xb 18 1F xb ff 18 1F xb ee ff 18 1F xb 18 1F xb ee ff	ORPW ORPWO OfRPWP OfIfrPW OfIPrPW	ORPW ORPWO OfRPWP OfIfrPW OfIPrPW	----	Δ Δ Δ Δ
EMUL	(D) × (Y) ⇒ Y:D 16 by 16 Bit Multiply (unsigned)	INH	13	ffO	ffO	----	Δ Δ – 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	----	Δ Δ – Δ
EORA # <i>opr8i</i> EORA <i>opr8a</i> EORA <i>opr16a</i> EORA <i>opr0_xysp</i> EORA <i>opr9_xysp</i> EORA <i>opr16_xysp</i> EORA [D,xysp] EORA [<i>opr16_xysp</i>]	(A) ⊕ (M) ⇒ A Exclusive-OR A with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	88 ii 98 dd B8 hh 1l A8 xb A8 xb ff A8 xb ee ff A8 xb A8 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rFP rOP rFP rPO frPP fIfrfP fIPrfP	----	Δ Δ 0 –

Notes:

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

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
EORB #opr8i EORB opr8a EORB opr16a EORB oprx0_xysp EORB oprx9_xysp EORB oprx16_xysp EORB [D,xysp] EORB [opr16,xysp]	$(B) \oplus (M) \Rightarrow B$ Exclusive-OR B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C8 ii D8 dd F8 hh 11 E8 xb E8 xb ff E8 xb ee ff E8 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	----	$\Delta \Delta 0 -$
ETBL oprx0_xysp	$(M:M+1) + [(B) \times ((M+2:M+3) - (M:M+1))] \Rightarrow D$ 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	ORRfffffFP ORRfffffFP	ORRfffffFP ORRfffffFP	----	$\Delta \Delta - \Delta$? C Bit is undefined in HC12
EXG abcdxys,abcdxys	$(r1) \Leftrightarrow (r2)$ (if r1 and r2 same size) or \$00:(r1) $\Rightarrow$ r2 (if r1=8-bit; r2=16-bit) or $(r1)_{low} \Leftrightarrow (r2)$ (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	$(D) \div (X) \Rightarrow X$; Remainder $\Rightarrow$ D 16 by 16 Bit Fractional Divide	INH	18 11	OffffffffffFO OffffffffffFO	OffffffffffFO OffffffffffFO	----	$-\Delta \Delta \Delta$
IBEQ abdxys,rel9	$(cnt) + 1 \Rightarrow cnt$ If $(cnt) = 0$, then Branch else Continue to next instruction Increment Counter and Branch if = 0 (cnt = A, B, D, X, Y, or SP)	REL (9-bit)	04 1b rr	PPP (branch) PPO (no branch)	PPP	----	----
IBNE abdxys,rel9	$(cnt) + 1 \Rightarrow cnt$ if $(cnt) \neq 0$, then Branch; else Continue to next instruction Increment Counter and Branch if $\neq 0$ (cnt = A, B, D, X, Y, or SP)	REL (9-bit)	04 1b rr	PPP (branch) PPO (no branch)	PPP	----	----
IDIV	$(D) \div (X) \Rightarrow X$; Remainder $\Rightarrow$ D 16 by 16 Bit Integer Divide (unsigned)	INH	18 10	OffffffffffFO OffffffffffFO	OffffffffffFO OffffffffffFO	----	$-\Delta 0 \Delta$
IDIVS	$(D) \div (X) \Rightarrow X$; Remainder $\Rightarrow$ D 16 by 16 Bit Integer Divide (signed)	INH	18 15	OffffffffffFO OffffffffffFO	OffffffffffFO OffffffffffFO	----	$\Delta \Delta \Delta \Delta$
INC opr16a INC oprx0_xysp INC oprx9_xysp INC oprx16_xysp INC [D,xysp] INC [opr16,xysp] INCA INCB	$(M) + \$01 \Rightarrow M$ Increment Memory Byte $(A) + \$01 \Rightarrow A$ Increment Acc. A $(B) + \$01 \Rightarrow B$ Increment Acc. B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	72 hh 11 62 xb 62 xb ff 62 xb ee ff 62 xb ee ff 42 52	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	$\Delta \Delta \Delta -$
INS	$(SP) + \$0001 \Rightarrow SP$ Translates to LEAS 1,SP	IDX	1B 81	Pf	pp ¹	----	----
INX	$(X) + \$0001 \Rightarrow X$ Increment Index Register X	INH	08	O	O	----	$-\Delta --$
INY	$(Y) + \$0001 \Rightarrow Y$ Increment Index Register Y	INH	02	O	O	----	$-\Delta --$
JMP opr16a JMP oprx0_xysp JMP oprx9_xysp JMP oprx16_xysp JMP [D,xysp] JMP [opr16,xysp]	Routine address $\Rightarrow$ PC Jump	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	06 hh 11 05 xb 05 xb ff 05 xb ee ff 05 xb 05 xb ee ff	PPP PPP PPP fPPP fIfrPPP fIfrPPP	PPP PPP PPP fPPP fIfrPPP fIfrPPP	----	----

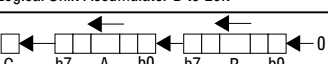
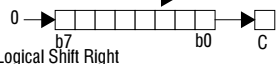
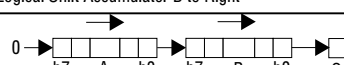
Note 1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
JSR <i>opr8a</i> JSR <i>opr16a</i> JSR <i>opr0_xysp</i> JSR <i>opr9_xysp</i> JSR <i>opr16_xysp</i> JSR [D, <i>xysp</i>] JSR [<i>opr16_xysp</i>]	(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 <i>rel16</i>	Long Branch if Carry Clear (if C = 0)	REL	18 24 qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBCS <i>rel16</i>	Long Branch if Carry Set (if C = 1)	REL	18 25 qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBEQ <i>rel16</i>	Long Branch if Equal (if Z = 1)	REL	18 27 qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBGE <i>rel16</i>	Long Branch Greater Than or Equal (if N ⊕ V = 0) (signed)	REL	18 2C qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBGT <i>rel16</i>	Long Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	18 2E qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBHI <i>rel16</i>	Long Branch if Higher (if C + Z = 0) (unsigned)	REL	18 22 qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBHS <i>rel16</i>	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 <i>rel16</i>	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 <i>rel16</i>	Long Branch if Lower (if C = 1) (unsigned) same function as LBCS	REL	18 25 qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBSL <i>rel16</i>	Long Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	18 23 qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBLT <i>rel16</i>	Long Branch if Less Than (if N ⊕ V = 1) (signed)	REL	18 2D qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBMI <i>rel16</i>	Long Branch if Minus (if N = 1)	REL	18 2B qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBNE <i>rel16</i>	Long Branch if Not Equal (if Z = 0)	REL	18 26 qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBPL <i>rel16</i>	Long Branch if Plus (if N = 0)	REL	18 2A qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBRA <i>rel16</i>	Long Branch Always (if 1=1)	REL	18 20 qq rr	OPPP	OPPP	----	----
LBRN <i>rel16</i>	Long Branch Never (if 1 = 0)	REL	18 21 qq rr	OPO	OPO	----	----
LBVC <i>rel16</i>	Long Branch if Overflow Bit Clear (if V=0)	REL	18 28 qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LBVS <i>rel16</i>	Long Branch if Overflow Bit Set (if V = 1)	REL	18 29 qq rr	OPPP/OPO ¹	OPPP/OPO ¹	----	----
LDAA # <i>opr8i</i> LDAA <i>opr8a</i> LDAA <i>opr16a</i> LDAA <i>opr0_xysp</i> LDAA <i>opr9_xysp</i> LDAA <i>opr16_xysp</i> LDAA [D, <i>xysp</i>] LDAA [<i>opr16_xysp</i>]	(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 rPf rOP rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ 0 –
LDAB # <i>opr8i</i> LDAB <i>opr8a</i> LDAB <i>opr16a</i> LDAB <i>opr0_xysp</i> LDAB <i>opr9_xysp</i> LDAB <i>opr16_xysp</i> LDAB [D, <i>xysp</i>] LDAB [<i>opr16_xysp</i>]	(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 rPf rOP rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ 0 –
LDD # <i>opr16i</i> LDD <i>opr8a</i> LDD <i>opr16a</i> LDD <i>opr0_xysp</i> LDD <i>opr9_xysp</i> LDD <i>opr16_xysp</i> LDD [D, <i>xysp</i>] LDD [<i>opr16_xysp</i>]	(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 fIfrPf fIPrPf	----	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.

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
LDS #opr16i LDS opr8a LDS opr16a LDS oprx0_xysp LDS oprx9_xysp LDS oprx16_xysp LDS [D,xysp] LDS [opr16,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 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 [opr16,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 ff	PO RPF RPO RPF RPO fRPP fIfRPF fIPRPF	OP RfP ROP RfP RPO fRPP fIfRfP fIPRfP	----	Δ Δ 0 -
LDY #opr16i LDY opr8a LDY opr16a LDY oprx0_xysp LDY oprx9_xysp LDY oprx16_xysp LDY [D,xysp] LDY [opr16,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 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 [opr16,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 ee ff 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 [opr16,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 ee ff 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 [opr16,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 OfrrPP OfIfrrPf OfIPrrPf	OrfP OrfPO OfrrPP OfIfrrfP OfIPrrfP	----	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.

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
MAXM <i>opr</i> x0_ <i>xy</i> sp MAXM <i>opr</i> x9_ <i>xy</i> sp MAXM <i>opr</i> x16_ <i>xy</i> sp MAXM [D, <i>xy</i> sp] MAXM [<i>opr</i> x16_ <i>xy</i> sp]	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 OfIfrPw OfIfrPw OfIPrPw	OrPw OrPwO OfIfrPw 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 <i>CPU12 Reference Manual</i> for special cases.	Special	01	RRfOw	RRfOw	--?–	????
MINA <i>opr</i> x0_ <i>xy</i> sp MINA <i>opr</i> x9_ <i>xy</i> sp MINA <i>opr</i> x16_ <i>xy</i> sp MINA [D, <i>xy</i> sp] MINA [<i>opr</i> x16_ <i>xy</i> sp]	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 OfIfrPf OfIfrPf OfIPrPf	OrPf OrPO OfIfrPf OfIfrPf OfIPrPf	----	Δ Δ Δ Δ
MINM <i>opr</i> x0_ <i>xy</i> sp MINM <i>opr</i> x9_ <i>xy</i> sp MINM <i>opr</i> x16_ <i>xy</i> sp MINM [D, <i>xy</i> sp] MINM [<i>opr</i> x16_ <i>xy</i> sp]	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 OfIfrPw OfIfrPw OfIPrPw	OrPw OrPwO OfIfrPw OfIfrPw OfIPrPw	----	Δ Δ Δ Δ
MOVB # <i>opr</i> 8, <i>opr</i> 16a ¹ MOVB # <i>opr</i> 8i, <i>opr</i> x0_ <i>xy</i> sp ¹ MOVB <i>opr</i> 16a, <i>opr</i> 16a ¹ MOVB <i>opr</i> 16a, <i>opr</i> x0_ <i>xy</i> sp ¹ MOVB <i>opr</i> x0_ <i>xy</i> sp, <i>opr</i> 16a ¹ MOVB <i>opr</i> x0_ <i>xy</i> sp, <i>opr</i> x0_ <i>xy</i> sp ¹	(M ₁) ⇒ 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 # <i>opr</i> x16, <i>opr</i> 16a ¹ MOVW # <i>opr</i> 16i, <i>opr</i> x0_ <i>xy</i> sp ¹ MOVW <i>opr</i> 16a, <i>opr</i> 16a ¹ MOVW <i>opr</i> 16a, <i>opr</i> x0_ <i>xy</i> sp ¹ MOVW <i>opr</i> x0_ <i>xy</i> sp, <i>opr</i> 16a ¹ MOVW <i>opr</i> x0_ <i>xy</i> sp, <i>opr</i> x0_ <i>xy</i> sp ¹	(M:M+1) ⇒ 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 <i>opr</i> 16a NEG <i>opr</i> x0_ <i>xy</i> sp NEG <i>opr</i> x9_ <i>xy</i> sp NEG <i>opr</i> x16_ <i>xy</i> sp NEG [D, <i>xy</i> sp] NEG [<i>opr</i> x16_ <i>xy</i> sp] NEGA NEGB	0 – (M) ⇒ M equivalent to (M) + 1 ⇒ M Two's Complement Negate 0 – (A) ⇒ A equivalent to (A) + 1 ⇒ A Negate Accumulator A 0 – (B) ⇒ B equivalent to (B) + 1 ⇒ 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	----	Δ Δ Δ Δ
NOP	No Operation	INH	A7	O	O	----	----
ORAA # <i>opr</i> 8i ORAA <i>opr</i> 8a ORAA <i>opr</i> 16a ORAA <i>opr</i> x0_ <i>xy</i> sp ORAA <i>opr</i> x9_ <i>xy</i> sp ORAA <i>opr</i> x16_ <i>xy</i> sp ORAA [D, <i>xy</i> sp] ORAA [<i>opr</i> x16_ <i>xy</i> sp]	(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 rPf rOP rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ 0 –

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

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
ORAB #opr8i ORAB opr8a ORAB opr16a ORAB oprx0_xysp ORAB oprx9_xysp ORAB oprx16_xysp ORAB [D,xysp] ORAB [opr16,xysp]	(B) + (M) $\Rightarrow$ 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 fIfPrPf fIPrPf	P rPf rOP rPf rPO frPP fIfPrPf fIPrPf	----	$\Delta \Delta 0 -$
ORCC #opr8i	(CCR) + M $\Rightarrow$ CCR Logical OR CCR with Memory	IMM	14 ii	P	P	$\uparrow - \uparrow \uparrow \uparrow$	$\uparrow \uparrow \uparrow \uparrow$
PSHA	(SP) - 1 $\Rightarrow$ SP; (A) $\Rightarrow$ M _(SP) Push Accumulator A onto Stack	INH	36	Os	Os	----	----
PSHB	(SP) - 1 $\Rightarrow$ SP; (B) $\Rightarrow$ M _(SP) Push Accumulator B onto Stack	INH	37	Os	Os	----	----
PSHC	(SP) - 1 $\Rightarrow$ SP; (CCR) $\Rightarrow$ M _(SP) Push CCR onto Stack	INH	39	Os	Os	----	----
PSHD	(SP) - 2 $\Rightarrow$ SP; (A:B) $\Rightarrow$ M _(SP) :M _(SP+1) Push D Accumulator onto Stack	INH	3B	OS	OS	----	----
PSHX	(SP) - 2 $\Rightarrow$ SP; (X _H :X _L) $\Rightarrow$ M _(SP) :M _(SP+1) Push Index Register X onto Stack	INH	34	OS	OS	----	----
PSHY	(SP) - 2 $\Rightarrow$ SP; (Y _H :Y _L) $\Rightarrow$ M _(SP) :M _(SP+1) Push Index Register Y onto Stack	INH	35	OS	OS	----	----
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	$\Delta \downarrow \Delta \Delta$	$\Delta \Delta \Delta \Delta$
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, ff + Orf(t,	Orf(t,tx)O (exit + re-entry replaces comma above if interrupted) ff + Orf(t,	--?-	??Δ?
REVV	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. REVV 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, fff + ORf(t,	ORf(t,Tx)O (loop to read weight if enabled) (r,RfRf) (exit + re-entry replaces comma above if interrupted) ffff + ORf(t, fff + ORf(t,	--?-	??Δ!

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

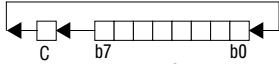
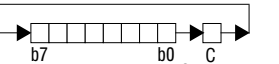
Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
ROL <i>opr16a</i> ROL <i>opr0_xysp</i> ROL <i>opr9_xysp</i> ROL <i>opr16_xysp</i> ROL [D, <i>xysp</i>] ROL [<i>opr16_xysp</i>] ROLA ROLB	 Rotate Memory Left through Carry Rotate A Left through Carry Rotate B Left through Carry	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	75 hh 11 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	---- ---- ---- ---- ---- ---- ---- ----	Δ Δ
ROR <i>opr16a</i> ROR <i>opr0_xysp</i> ROR <i>opr9_xysp</i> ROR <i>opr16_xysp</i> ROR [D, <i>xysp</i>] ROR [<i>opr16_xysp</i>] 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 11 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	---- ---- ---- ---- ---- ---- ----	Δ Δ
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 uUUUUvfPPP	Δ ↓ Δ Δ Δ Δ Δ Δ	Δ Δ Δ Δ Δ Δ Δ Δ
RTS	$(M_{(SP)}; M_{(SP+1)} \Rightarrow PC_H; PC_L;$ $(SP) + 2 \Rightarrow SP$ Return from Subroutine	INH	3D	UfPPP	UfPPP	----	----
SBA	$(A) - (B) \Rightarrow A$ Subtract B from A	INH	18 16	OO	OO	----	Δ Δ Δ Δ
SBCA # <i>opr8i</i> SBCA <i>opr8a</i> SBCA <i>opr16a</i> SBCA <i>opr0_xysp</i> SBCA <i>opr9_xysp</i> SBCA <i>opr16_xysp</i> SBCA [D, <i>xysp</i>] SBCA [<i>opr16_xysp</i>]	$(A) - (M) - C \Rightarrow A$ Subtract with Borrow from A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	82 ii 92 dd B2 hh 11 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
SBCB # <i>opr8i</i> SBCB <i>opr8a</i> SBCB <i>opr16a</i> SBCB <i>opr0_xysp</i> SBCB <i>opr9_xysp</i> SBCB <i>opr16_xysp</i> SBCB [D, <i>xysp</i>] SBCB [<i>opr16_xysp</i>]	$(B) - (M) - C \Rightarrow B$ Subtract with Borrow from B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C2 ii D2 dd F2 hh 11 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
SEC	$1 \Rightarrow C$ Translates to ORCC #01	IMM	14 01	P	P	----	---1
SEI	$1 \Rightarrow I$; (inhibit I interrupts) Translates to ORCC #10	IMM	14 10	P	P	---1	----
SEV	$1 \Rightarrow V$ Translates to ORCC #02	IMM	14 02	P	P	----	--1-
SEX <i>abc,dxys</i>	$\$00:(r1) \Rightarrow r2$ if r1, bit 7 is 0 or $\$FF:(r1) \Rightarrow r2$ 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	----	----

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
STAA <i>opr8a</i> STAA <i>opr16a</i> STAA <i>opr0_xysp</i> STAA <i>opr9_xysp</i> STAA <i>opr16_xysp</i> STAA [D, <i>xysp</i>] STAA [<i>opr16_xysp</i>]	(A) $\Rightarrow$ M Store Accumulator A to Memory	DIR EXT IDX IDX1 IDX2 [D, IDX] [IDX2]	5A dd 7A hh 1l 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	----	$\Delta \Delta 0 -$
STAB <i>opr8a</i> STAB <i>opr16a</i> STAB <i>opr0_xysp</i> STAB <i>opr9_xysp</i> STAB <i>opr16_xysp</i> STAB [D, <i>xysp</i>] STAB [<i>opr16_xysp</i>]	(B) $\Rightarrow$ M Store Accumulator B to Memory	DIR EXT IDX IDX1 IDX2 [D, IDX] [IDX2]	5B dd 7B hh 1l 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	----	$\Delta \Delta 0 -$
STD <i>opr8a</i> STD <i>opr16a</i> STD <i>opr0_xysp</i> STD <i>opr9_xysp</i> STD <i>opr16_xysp</i> STD [D, <i>xysp</i>] STD [<i>opr16_xysp</i>]	(A) $\Rightarrow$ M, (B) $\Rightarrow$ M+1 Store Double Accumulator	DIR EXT IDX IDX1 IDX2 [D, IDX] [IDX2]	5C dd 7C hh 1l 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	----	$\Delta \Delta 0 -$
STOP	(SP) - 2 $\Rightarrow$ SP; RTN _H :RTN _L $\Rightarrow$ M _(SP) :M _(SP+1) ; (SP) - 2 $\Rightarrow$ SP; (Y _H :Y _L) $\Rightarrow$ M _(SP) :M _(SP+1) ; (SP) - 2 $\Rightarrow$ SP; (X _H :X _L) $\Rightarrow$ M _(SP) :M _(SP+1) ; (SP) - 2 $\Rightarrow$ SP; (B:A) $\Rightarrow$ M _(SP) :M _(SP+1) ; (SP) - 1 $\Rightarrow$ SP; (CCR) $\Rightarrow$ 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	----	----
STS <i>opr8a</i> STS <i>opr16a</i> STS <i>opr0_xysp</i> STS <i>opr9_xysp</i> STS <i>opr16_xysp</i> STS [D, <i>xysp</i>] STS [<i>opr16_xysp</i>]	(SP _H :SP _L) $\Rightarrow$ M:M+1 Store Stack Pointer	DIR EXT IDX IDX1 IDX2 [D, IDX] [IDX2]	5F dd 7F hh 1l 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	----	$\Delta \Delta 0 -$
STX <i>opr8a</i> STX <i>opr16a</i> STX <i>opr0_xysp</i> STX <i>opr9_xysp</i> STX <i>opr16_xysp</i> STX [D, <i>xysp</i>] STX [<i>opr16_xysp</i>]	(X _H :X _L) $\Rightarrow$ M:M+1 Store Index Register X	DIR EXT IDX IDX1 IDX2 [D, IDX] [IDX2]	5E dd 7E hh 1l 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	----	$\Delta \Delta 0 -$
STY <i>opr8a</i> STY <i>opr16a</i> STY <i>opr0_xysp</i> STY <i>opr9_xysp</i> STY <i>opr16_xysp</i> STY [D, <i>xysp</i>] STY [<i>opr16_xysp</i>]	(Y _H :Y _L) $\Rightarrow$ M:M+1 Store Index Register Y	DIR EXT IDX IDX1 IDX2 [D, IDX] [IDX2]	5D dd 7D hh 1l 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	----	$\Delta \Delta 0 -$
SUBA # <i>opr8i</i> SUBA <i>opr8a</i> SUBA <i>opr16a</i> SUBA <i>opr0_xysp</i> SUBA <i>opr9_xysp</i> SUBA <i>opr16_xysp</i> SUBA [D, <i>xysp</i>] SUBA [<i>opr16_xysp</i>]	(A) - (M) $\Rightarrow$ A Subtract Memory from Accumulator A	IMM DIR EXT IDX IDX1 IDX2 [D, IDX] [IDX2]	80 ii 90 dd B0 hh 1l A0 xb A0 xb ff A0 xb ee ff A0 xb A0 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rOP rPf rPO frPP fIfrPf fIPrPf	----	$\Delta \Delta \Delta \Delta$

Instruction Reference

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
SUBB #opr <i>8i</i> SUBB opr <i>8a</i> SUBB opr <i>16a</i> SUBB opr <i>0_xysp</i> SUBB opr <i>9_xysp</i> SUBB opr <i>16_xysp</i> SUBB [D, <i>xysp</i>] SUBB [opr <i>16_xysp</i>]	(B) – (M) ⇒ 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 #opr <i>16i</i> SUBD opr <i>8a</i> SUBD opr <i>16a</i> SUBD opr <i>0_xysp</i> SUBD opr <i>9_xysp</i> SUBD opr <i>16_xysp</i> SUBD [D, <i>xysp</i>] SUBD [opr <i>16_xysp</i>]	(D) – (M:M+1) ⇒ 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 ⇒ 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; (SWI Vector) ⇒ PC Software Interrupt	INH	3F	VSPSSPSsP* (for Reset) VfPPP	VSPSSPSsP* VfPPP	----1 11–1	---- ----
*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 <i>abdxys,rel9</i>	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 1b rr	PPP (branch) PPO (no branch)	PPP	----	----
TBL opr <i>0_xysp</i>	(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	ORffffP OrrffffP	OrffffP	----	Δ Δ – Δ ? C Bit is undefined in HC12
TBNE <i>abdxys,rel9</i>	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 1b rr	PPP (branch) PPO (no branch)	PPP	----	----
TFR <i>abcdxys,abcdxys</i>	(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	---- Δ ↓ Δ Δ	---- Δ Δ Δ Δ
TPA	(CCR) ⇒ A Translates to TFR CCR, A	INH	B7 20	P	P	----	----

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	M68HC12		
TRAP <i>trapnum</i>	$(SP) - 2 \Rightarrow SP;$ $RTN_H:RTN_L \Rightarrow M_{(SP)}:M_{(SP+1)};$ $(SP) - 2 \Rightarrow SP; (Y_H:Y_L) \Rightarrow M_{(SP)}:M_{(SP+1)};$ $(SP) - 2 \Rightarrow SP; (X_H:X_L) \Rightarrow M_{(SP)}:M_{(SP+1)};$ $(SP) - 2 \Rightarrow SP; (B:A) \Rightarrow M_{(SP)}:M_{(SP+1)};$ $(SP) - 1 \Rightarrow SP; (CCR) \Rightarrow M_{(SP)}$ $1 \Rightarrow I; (TRAP\ Vector) \Rightarrow PC$ Unimplemented opcode trap	INH	18 tn tn = \$30-\$39 or \$40-\$FF	OVSPSSPSSp	OVSPSSPSSp	---1	----
TST <i>opr16a</i> TST <i>oprX0,xysp</i> TST <i>oprX9,xysp</i> TST <i>oprX16,xysp</i> 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 ee ff 97 D7	rPO rPf rPO frPP fIfrPf fIPrPf O O	rOP rfP rPO frPP fIfrfP fIPrfP O O	----	$\Delta \Delta 00$
TSX	$(SP) \Rightarrow X$ Translates to TFR SP,X	INH	B7 75	P	P	----	----
TSY	$(SP) \Rightarrow Y$ Translates to TFR SP,Y	INH	B7 76	P	P	----	----
TXS	$(X) \Rightarrow SP$ Translates to TFR X,SP	INH	B7 57	P	P	----	----
TYS	$(Y) \Rightarrow SP$ Translates to TFR Y,SP	INH	B7 67	P	P	----	----
WAI	$(SP) - 2 \Rightarrow SP;$ $RTN_H:RTN_L \Rightarrow M_{(SP)}:M_{(SP+1)};$ $(SP) - 2 \Rightarrow SP; (Y_H:Y_L) \Rightarrow M_{(SP)}:M_{(SP+1)};$ $(SP) - 2 \Rightarrow SP; (X_H:X_L) \Rightarrow M_{(SP)}:M_{(SP+1)};$ $(SP) - 2 \Rightarrow SP; (B:A) \Rightarrow M_{(SP)}:M_{(SP+1)};$ $(SP) - 1 \Rightarrow SP; (CCR) \Rightarrow M_{(SP)};$ WAIT for interrupt	INH	3E	OSSSSsf (after interrupt) fVfPPP	OSSsfSsf VfPPP	---- or ---1 or -1-1	---- ---- ----
WAV	$B \sum_{i=1} S_i F_i \Rightarrow Y:D \quad \text{and} \quad \sum_{i=1}^B F_i \Rightarrow X$ 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,ffff)O (add if interrupt) SSS + UUUrr, SSSf + UUUrr		--?-	?Δ??
wav 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	$(D) \Leftrightarrow (X)$ Translates to EXG D, X	INH	B7 C5	P	P	----	----
XGDY	$(D) \Leftrightarrow (Y)$ Translates to EXG D, Y	INH	B7 C6	P	P	----	----