

Notation Used in Instruction Set Summary

CPU Register Notation

Accumulator A — A or a	Index Register Y — Y or y
Accumulator B — B or b	Stack Pointer — SP, sp, or s
Accumulator D — D or d	Program Counter — PC, pc, or p
Index Register X — X or x	Condition Code Register — CCR or c

Explanation of Italic Expressions in Source Form Column

abc — A or B or CCR
abcdxys — A or B or CCR or D or X or Y or SP. Some assemblers also allow T2 or T3.
abd — A or B or D
abdxys — A or B or D or X or Y or SP
dxys — D or X or Y or SP
msk8 — 8-bit mask, some assemblers require # symbol before value
opr8i — 8-bit immediate value
opr16i — 16-bit immediate value
opr8a — 8-bit address used with direct address mode
opr16a — 16-bit address value
oprX0_xysp — Indexed addressing postbyte code:
oprX3,-xys Predecrement X or Y or SP by 1 . . . 8
oprX3,+xys Preincrement X or Y or SP by 1 . . . 8
oprX3,xys- Postdecrement X or Y or SP by 1 . . . 8
oprX3,xys+ Postincrement X or Y or SP by 1 . . . 8
opr5,xysp 5-bit constant offset from X or Y or SP or PC
abd,xysp Accumulator A or B or D offset from X or Y or SP or PC
oprX3 — Any positive integer 1 . . . 8 for pre/post increment/decrement
oprX5 — Any integer in the range -16 . . . +15
oprX9 — Any integer in the range -256 . . . +255
oprX16 — Any integer in the range -32,768 . . . 65,535
page — 8-bit value for PPAGE, some assemblers require # symbol before this value
rel8 — Label of branch destination within -256 to +255 locations
rel9 — Label of branch destination within -512 to +511 locations
rel16 — Any label within 64K memory space
trapnum — Any 8-bit integer in the range \$30-\$39 or \$40-\$FF
xys — X or Y or SP
xysp — X or Y or SP or PC

Operators

+ — Addition
- — Subtraction
• — Logical AND
+ — Logical OR (inclusive)
⊕ — Logical exclusive OR
× — Multiplication
÷ — Division
 \bar{M} — Negation. One's complement (invert each bit of M)
: — Concatenate
Example: A : B means the 16-bit value formed by concatenating 8-bit accumulator A with 8-bit accumulator B. A is in the high-order position.

Operators (continued)

⇒ — Transfer
Example: (A) ⇒ M means the content of accumulator A is transferred to memory location M.
⇔ — Exchange
Example: D ⇔ X means exchange the contents of D with those of X.

Address Mode Notation

INH — Inherent; no operands in object code
IMM — Immediate; operand in object code
DIR — Direct; operand is the lower byte of an address from \$0000 to \$00FF
EXT — Operand is a 16-bit address
REL — Two's complement relative offset; for branch instructions
IDX — Indexed (no extension bytes); includes:
5-bit constant offset from X, Y, SP, or PC
Pre/post increment/decrement by 1 . . . 8
Accumulator A, B, or D offset
IDX1 — 9-bit signed offset from X, Y, SP, or PC; 1 extension byte
IDX2 — 16-bit signed offset from X, Y, SP, or PC; 2 extension bytes
[IDX2] — Indexed-indirect; 16-bit offset from X, Y, SP, or PC
[D, IDX] — Indexed-indirect; accumulator D offset from X, Y, SP, or PC

Machine Coding

dd — 8-bit direct address \$0000 to \$00FF. (High byte assumed to be \$00).
ee — High-order byte of a 16-bit constant offset for indexed addressing.
eb — Exchange/Transfer post-byte.
ff — Low-order eight bits of a 9-bit signed constant offset for indexed addressing, or low-order byte of a 16-bit constant offset for indexed addressing.
hh — High-order byte of a 16-bit extended address.
ii — 8-bit immediate data value.
jj — High-order byte of a 16-bit immediate data value.
kk — Low-order byte of a 16-bit immediate data value.
lb — Loop primitive (DBNE) post-byte.
ll — Low-order byte of a 16-bit extended address.
mm — 8-bit immediate mask value for bit manipulation instructions.
Set bits indicate bits to be affected.
pg — Program page (bank) number used in CALL instruction.
qq — High-order byte of a 16-bit relative offset for long branches.
tn — Trap number \$30-\$39 or \$40-\$FF.
rr — Signed relative offset \$80 (-128) to \$7F (+127).
Offset relative to the byte following the relative offset byte, or low-order byte of a 16-bit relative offset for long branches.
xb — Indexed addressing post-byte.

Access Detail

Each code letter except (,) and comma equals one CPU cycle. Uppercase = 16-bit operation and lowercase = 8-bit operation.

f — Free cycle, CPU doesn't use bus
g — Read PPAGE internally
I — Read indirect pointer (indexed indirect)
i — Read indirect PPAGE value (CALL indirect only)
n — Write PPAGE internally
O — Optional program word fetch (P) if instruction is misaligned and has an odd number of bytes of object code — otherwise, appears as a free cycle (f); Page 2 prebyte treated as a separate 1-byte instruction
P — Program word fetch (always an aligned-word read)
r — 8-bit data read
R — 16-bit data read
s — 8-bit stack write
S — 16-bit stack write
w — 8-bit data write
W — 16-bit data write
u — 8-bit stack read
U — 16-bit stack read
V — 16-bit vector fetch (always an aligned-word read)
t — 8-bit conditional read (or free cycle)
T — 16-bit conditional read (or free cycle)
x — 8-bit conditional write (or free cycle)
() — Indicate a microcode loop
, — Indicates where an interrupt could be honored

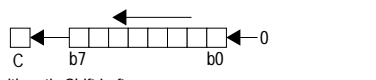
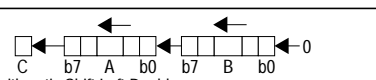
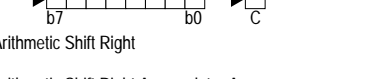
Special Cases

PPP/P — Short branch, PPP if branch taken, P if not
OPPP/OPO — Long branch, OPPP if branch taken, OPO if not

Condition Codes Columns

— — Status bit not affected by operation.
0 — Status bit cleared by operation.
1 — Status bit set by operation.
Δ — Status bit affected by operation.
fl — Status bit may be cleared or remain set, but is not set by operation.
↑ — Status bit may be set or remain cleared, but is not cleared by operation.
? — Status bit may be changed by operation but the final state is not defined.
! — Status bit used for a special purpose.

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
ABA	(A) + (B) ⇒ A Add Accumulators A and B	INH	18 06	OO	--Δ-	ΔΔΔΔ
ABX	(B) + (X) ⇒ X Translates to LEAX B,X	IDX	1A E5	Pf	----	----
ABY	(B) + (Y) ⇒ Y Translates to LEAY B,Y	IDX	19 ED	Pf	----	----
ADCA #opr8i ADCA opr8a ADCA opr16a ADCA oprx0_xyssp ADCA oprx9_xyssp ADCA oprx16_xyssp ADCA [D,xyssp] ADCA [opr16,xyssp]	(A) + (M) + C ⇒ 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	--Δ-	ΔΔΔΔ
ADCB #opr8i ADCB opr8a ADCB opr16a ADCB oprx0_xyssp ADCB oprx9_xyssp ADCB oprx16_xyssp ADCB [D,xyssp] ADCB [opr16,xyssp]	(B) + (M) + C ⇒ B Add with Carry to B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	D9 ii C9 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	--Δ-	ΔΔΔΔ
ADDA #opr8i ADDA opr8a ADDA opr16a ADDA oprx0_xyssp ADDA oprx9_xyssp ADDA oprx16_xyssp ADDA [D,xyssp] ADDA [opr16,xyssp]	(A) + (M) ⇒ 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	--Δ-	ΔΔΔΔ
ADDB #opr8i ADDB opr8a ADDB opr16a ADDB oprx0_xyssp ADDB oprx9_xyssp ADDB oprx16_xyssp ADDB [D,xyssp] ADDB [opr16,xyssp]	(B) + (M) ⇒ 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	--Δ-	ΔΔΔΔ
ADDD #opr16i ADDD opr8a ADDD opr16a ADDD oprx0_xyssp ADDD oprx9_xyssp ADDD oprx16_xyssp ADDD [D,xyssp] ADDD [opr16,xyssp]	(A:B) + (M:M+1) ⇒ 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	----	ΔΔΔΔ
ANDA #opr8i ANDA opr8a ANDA opr16a ANDA oprx0_xyssp ANDA oprx9_xyssp ANDA oprx16_xyssp ANDA [D,xyssp] ANDA [opr16,xyssp]	(A) • (M) ⇒ 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	----	ΔΔ0-
ANDB #opr8i ANDB opr8a ANDB opr16a ANDB oprx0_xyssp ANDB oprx9_xyssp ANDB oprx16_xyssp ANDB [D,xyssp] ANDB [opr16,xyssp]	(B) • (M) ⇒ 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	----	ΔΔ0-
ANDCC #opr8i	(CCR) • (M) ⇒ CCR Logical AND CCR with Memory	IMM	10 ii	P	↓↓↓↓	↓↓↓↓

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
ASL opr16a ASL oprx0_xyssp ASL oprx9_xyssp ASL oprx16_xyssp ASL [D,xyssp] ASL [opr16,xyssp] ASLA ASLB	 <p>Arithmetic Shift Left</p> <p>Arithmetic Shift Left Accumulator A</p> <p>Arithmetic Shift Left Accumulator B</p>	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	78 hh 11 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	----	ΔΔΔΔ
ASLD	 <p>Arithmetic Shift Left Double</p>	INH	59	O	----	ΔΔΔΔ
ASR opr16a ASR oprx0_xyssp ASR oprx9_xyssp ASR oprx16_xyssp ASR [D,xyssp] ASR [opr16,xyssp] ASRA ASRB	 <p>Arithmetic Shift Right</p> <p>Arithmetic Shift Right Accumulator A</p> <p>Arithmetic Shift Right Accumulator B</p>	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	77 hh 11 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	----	ΔΔΔΔ
BCC rel8	Branch if Carry Clear (if C = 0)	REL	24 rr	PPP/P ¹	----	----
BCLR opr8a, msk8 BCLR opr16a, msk8 BCLR oprx0_xyssp, msk8 BCLR oprx9_xyssp, msk8 BCLR oprx16_xyssp, msk8	(M) • (mm) ⇒ M Clear Bit(s) in Memory	DIR EXT IDX IDX1 IDX2	4D dd mm 1D hh 11 mm 0D xb mm 0D xb ff mm 0D xb ee ff mm	rPwO rPwP rPwO rPwP frPwPO	----	ΔΔ0-
BCS rel8	Branch if Carry Set (if C = 1)	REL	25 rr	PPP/P ¹	----	----
BEQ rel8	Branch if Equal (if Z = 1)	REL	27 rr	PPP/P ¹	----	----
BGE rel8	Branch if Greater Than or Equal (if N ⊕ V = 0) (signed)	REL	2C rr	PPP/P ¹	----	----
BGND	Place CPU in Background Mode see CPU12 Reference Manual	INH	00	VfPPP	----	----
BGT rel8	Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	2E rr	PPP/P ¹	----	----
BHI rel8	Branch if Higher (if C + Z = 0) (unsigned)	REL	22 rr	PPP/P ¹	----	----
BHS rel8	Branch if Higher or Same (if C = 0) (unsigned) same function as BCC	REL	24 rr	PPP/P ¹	----	----
BITA #opr8i BITA opr8a BITA opr16a BITA oprx0_xyssp BITA oprx9_xyssp BITA oprx16_xyssp BITA [D,xyssp] BITA [opr16,xyssp]	(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 11 A5 xb A5 xb ff A5 xb ee ff A5 xb A5 xb ee ff	P rPf rPO rPf rPO frPP fIFrPf fIPrPf	----	ΔΔ0-
BITB #opr8i BITB opr8a BITB opr16a BITB oprx0_xyssp BITB oprx9_xyssp BITB oprx16_xyssp BITB [D,xyssp] BITB [opr16,xyssp]	(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 11 E5 xb E5 xb ff E5 xb ee ff E5 xb E5 xb ee ff	P rPf rPO rPf rPO frPP fIFrPf fIPrPf	----	ΔΔ0-
BLE rel8	Branch if Less Than or Equal (if Z + (N ⊕ V) = 1) (signed)	REL	2F rr	PPP/P ¹	----	----
BLO rel8	Branch if Lower (if C = 1) (unsigned) same function as BCS	REL	25 rr	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.

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
BLS <i>rel8</i>	Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	23 rr	PPP/P ¹	----	----
BLT <i>rel8</i>	Branch if Less Than (if N ⊕ V = 1) (signed)	REL	2D rr	PPP/P ¹	----	----
BMI <i>rel8</i>	Branch if Minus (if N = 1)	REL	2B rr	PPP/P ¹	----	----
BNE <i>rel8</i>	Branch if Not Equal (if Z = 0)	REL	26 rr	PPP/P ¹	----	----
BPL <i>rel8</i>	Branch if Plus (if N = 0)	REL	2A rr	PPP/P ¹	----	----
BRA <i>rel8</i>	Branch Always (if 1 = 1)	REL	20 rr	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	----	----
BRN <i>rel8</i>	Branch Never (if 1 = 0)	REL	21 rr	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 (M̄) • (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	----	----
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	----	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	----	----
BVC <i>rel8</i>	Branch if Overflow Bit Clear (if V = 0)	REL	28 rr	PPP/P ¹	----	----
BVS <i>rel8</i>	Branch if Overflow Bit Set (if V = 1)	REL	29 rr	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	----	----
CBA	(A) - (B) Compare 8-Bit Accumulators	INH	18 17	OO	----	Δ Δ Δ Δ
CLC	0 ⇒ C <i>Translates to ANDCC # \$FE</i>	IMM	10 FE	P	----	--- 0
CLI	0 ⇒ I <i>Translates to ANDCC # \$EF</i> (enables I-bit interrupts)	IMM	10 EF	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	----	0 1 0 0
CLV	0 ⇒ V <i>Translates to ANDCC # \$FD</i>	IMM	10 FD	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.

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
CMPA # <i>opr8i</i> CMPA <i>opr8a</i> CMPA <i>opr16a</i> CMPA <i>opr0_xysp</i> CMPA <i>opr9_xysp</i> CMPA <i>opr16_xysp</i> CMPA [D, <i>xysp</i>] CMPA [<i>opr16, xysp</i>]	(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	----	Δ Δ Δ Δ
CMPB # <i>opr8i</i> CMPB <i>opr8a</i> CMPB <i>opr16a</i> CMPB <i>opr0_xysp</i> CMPB <i>opr9_xysp</i> CMPB <i>opr16_xysp</i> CMPB [D, <i>xysp</i>] CMPB [<i>opr16, xysp</i>]	(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	----	Δ Δ Δ Δ
COM <i>opr16a</i> COM <i>opr0_xysp</i> COM <i>opr9_xysp</i> COM <i>opr16_xysp</i> COM [D, <i>xysp</i>] COM [<i>opr16, xysp</i>] COMA COMB	(M̄) ⇒ M equivalent to \$FF - (M) ⇒ M 1's Complement Memory Location (Ā) ⇒ A Complement Accumulator A (B̄) ⇒ 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	----	Δ Δ 0 1
CPD # <i>opr16i</i> CPD <i>opr8a</i> CPD <i>opr16a</i> CPD <i>opr0_xysp</i> CPD <i>opr9_xysp</i> CPD <i>opr16_xysp</i> CPD [D, <i>xysp</i>] CPD [<i>opr16, xysp</i>]	(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	----	Δ Δ Δ Δ
CPS # <i>opr16i</i> CPS <i>opr8a</i> CPS <i>opr16a</i> CPS <i>opr0_xysp</i> CPS <i>opr9_xysp</i> CPS <i>opr16_xysp</i> CPS [D, <i>xysp</i>] CPS [<i>opr16, xysp</i>]	(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	----	Δ Δ Δ Δ
CPX # <i>opr16i</i> CPX <i>opr8a</i> CPX <i>opr16a</i> CPX <i>opr0_xysp</i> CPX <i>opr9_xysp</i> CPX <i>opr16_xysp</i> CPX [D, <i>xysp</i>] CPX [<i>opr16, xysp</i>]	(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	----	Δ Δ Δ Δ
CPY # <i>opr16i</i> CPY <i>opr8a</i> CPY <i>opr16a</i> CPY <i>opr0_xysp</i> CPY <i>opr9_xysp</i> CPY <i>opr16_xysp</i> CPY [D, <i>xysp</i>] CPY [<i>opr16, xysp</i>]	(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	----	Δ Δ Δ Δ
DAA	Adjust Sum to BCD Decimal Adjust Accumulator A	INH	18 07	Ofo	----	Δ Δ ? Δ
DBEQ <i>abdxy, rel9</i>	(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)	----	----

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
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)	----	----
DEC <i>opr16a</i> DEC <i>opr0_xysp</i> DEC <i>opr9_xysp</i> DEC <i>opr16_xysp</i> DEC [D, <i>xysp</i>] 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 11 63 xb 63 xb ff 63 xb ee ff 63 xb ee ff 43 53	rPwO rPw rPwO frPwP fIFrPw fIPrPw O O	----	Δ Δ Δ -
DES	(SP) - \$0001 ⇒ SP <i>Translates to LEAS -1,SP</i>	IDX	1B 9F	Pf	----	----
DEX	(X) - \$0001 ⇒ X Decrement Index Register X	INH	09	O	----	- Δ - -
DEY	(Y) - \$0001 ⇒ Y Decrement Index Register Y	INH	03	O	----	- Δ - -
EDIV	(Y:D) ÷ (X) ⇒ Y Remainder ⇒ D 32 by 16 Bit ⇒ 16 Bit Divide (unsigned)	INH	11	fffffffffffo	----	Δ Δ Δ Δ
EDIVS	(Y:D) ÷ (X) ⇒ Y Remainder ⇒ D 32 by 16 Bit ⇒ 16 Bit Divide (signed)	INH	18 14	offfffffffo	----	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 11	ORROffRRfWwP	----	Δ Δ Δ Δ
EMAXD <i>opr0_xysp</i> EMAXD <i>opr9_xysp</i> EMAXD <i>opr16_xysp</i> EMAXD [D, <i>xysp</i>] 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 OfRRPp OfIFRPf OfIPRPf	----	Δ Δ Δ Δ
EMAXM <i>opr0_xysp</i> EMAXM <i>opr9_xysp</i> EMAXM <i>opr16_xysp</i> EMAXM [D, <i>xysp</i>] 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	----	Δ Δ Δ Δ
EMIND <i>opr0_xysp</i> EMIND <i>opr9_xysp</i> EMIND <i>opr16_xysp</i> EMIND [D, <i>xysp</i>] 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 OfRRPp OfIFRPf OfIPRPf	----	Δ Δ Δ Δ
EMINM <i>opr0_xysp</i> EMINM <i>opr9_xysp</i> EMINM <i>opr16_xysp</i> EMINM [D, <i>xysp</i>] 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	----	Δ Δ Δ Δ
EMUL	(D) × (Y) ⇒ Y:D 16 by 16 Bit Multiply (unsigned)	INH	13	ffo	----	Δ Δ - Δ
EMULS	(D) × (Y) ⇒ Y:D 16 by 16 Bit Multiply (signed)	INH	18 13	offo (if followed by page 2 instruction) offo	----	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, <i>xysp</i>] 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 11 A8 xb A8 xb ff A8 xb ee ff A8 xb A8 xb ee ff	P rPf rPO rPf rPO frPP fIFrPf fIPrPf	----	Δ Δ 0 -

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
EORB # <i>opr8i</i> EORB <i>opr8a</i> EORB <i>opr16a</i> EORB <i>opr0_xysp</i> EORB <i>opr9_xysp</i> EORB <i>opr16_xysp</i> EORB [D, <i>xysp</i>] EORB [<i>opr16_xysp</i>]	(B) ⊕ (M) ⇒ 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 E8 xb ee ff	P rPf rPO rPf rPO frPP fIFrPf fIPrPf	----	Δ Δ 0 -
ETBL <i>opr0_xysp</i>	(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	ORRffffffP	----	Δ Δ - Δ C Bit is undefined in HC12
EXG <i>abcdxys, abcdxys</i>	(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	----	----
FDIV	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Fractional Divide	INH	18 11	offfffffffo	----	- Δ Δ Δ
IBEQ <i>abdxys, rel9</i>	(cnt) + 1 ⇒ 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)	----	----
IBNE <i>abdxys, rel9</i>	(cnt) + 1 ⇒ cnt if (cnt) not = 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)	----	----
IDIV	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Integer Divide (unsigned)	INH	18 10	offfffffffo	----	- Δ 0 Δ
IDIVS	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Integer Divide (signed)	INH	18 15	offfffffffo	----	Δ Δ Δ Δ
INC <i>opr16a</i> INC <i>opr0_xysp</i> INC <i>opr9_xysp</i> INC <i>opr16_xysp</i> INC [D, <i>xysp</i>] INC [<i>opr16_xysp</i>] 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 11 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	----	Δ Δ Δ -
INS	(SP) + \$0001 ⇒ SP <i>Translates to LEAS 1,SP</i>	IDX	1B 81	Pf	----	----
INX	(X) + \$0001 ⇒ X Increment Index Register X	INH	08	O	----	- Δ - -
INY	(Y) + \$0001 ⇒ Y Increment Index Register Y	INH	02	O	----	- Δ - -
JMP <i>opr16a</i> JMP <i>opr0_xysp</i> JMP <i>opr9_xysp</i> JMP <i>opr16_xysp</i> JMP [D, <i>xysp</i>] JMP [<i>opr16_xysp</i>]	Routine address ⇒ 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 fIFPPP fIFPPP	----	----

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

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.

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
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	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	17 dd 16 hh 11 15 xb 15 xb ff 15 xb ee ff 15 xb 15 xb ee ff	SPPP SPPP 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	----	----
LBCS <i>rel16</i>	Long Branch if Carry Set (if C = 1)	REL	18 25 qq rr	OPPP/OPO ¹	----	----
LB EQ <i>rel16</i>	Long Branch if Equal (if Z = 1)	REL	18 27 qq rr	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	----	----
LBGT <i>rel16</i>	Long Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	18 2E qq rr	OPPP/OPO ¹	----	----
LBHI <i>rel16</i>	Long Branch if Higher (if C + Z = 0) (unsigned)	REL	18 22 qq rr	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	----	----
LBL E <i>rel16</i>	Long Branch if Less Than or Equal (if Z + (N ⊕ V) = 1) (signed)	REL	18 2F qq rr	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	----	----
LBLS <i>rel16</i>	Long Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	18 23 qq rr	OPPP/OPO ¹	----	----
LB LT <i>rel16</i>	Long Branch if Less Than (if N ⊕ V = 1) (signed)	REL	18 2D qq rr	OPPP/OPO ¹	----	----
LBMI <i>rel16</i>	Long Branch if Minus (if N = 1)	REL	18 2B qq rr	OPPP/OPO ¹	----	----
LBNE <i>rel16</i>	Long Branch if Not Equal (if Z = 0)	REL	18 26 qq rr	OPPP/OPO ¹	----	----
LBPL <i>rel16</i>	Long Branch if Plus (if N = 0)	REL	18 2A qq rr	OPPP/OPO ¹	----	----
LBRA <i>rel16</i>	Long Branch Always (if 1=1)	REL	18 20 qq rr	OPPP	----	----
LB RN <i>rel16</i>	Long Branch Never (if 1 = 0)	REL	18 21 qq rr	OPO	----	----
LBVC <i>rel16</i>	Long Branch if Overflow Bit Clear (if V=0)	REL	18 28 qq rr	OPPP/OPO ¹	----	----
LBVS <i>rel16</i>	Long Branch if Overflow Bit Set (if V = 1)	REL	18 29 qq rr	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 11 A6 xb A6 xb ff A6 xb ee ff A6 xb A6 xb ee ff	P rPf rPO 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 11 E6 xb E6 xb ff E6 xb ee ff E6 xb E6 xb ee ff	P rPf rPO 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 11 EC xb EC xb ff EC xb ee ff EC xb EC xb ee ff	PO RPF RPO RPF 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.

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
LDS # <i>opr16i</i> LDS <i>opr8a</i> LDS <i>opr16a</i> LDS <i>opr0_xysp</i> LDS <i>opr9_xysp</i> LDS <i>opr16_xysp</i> LDS [D, <i>xysp</i>] LDS [<i>opr16_xysp</i>]	(M:M+1) ⇒ SP Load Stack Pointer	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CF jj kk DF dd FF hh 11 EF xb EF xb ff EF xb ee ff EF xb EF xb ee ff	PO RPF RPO RPF RPO frPP fIFrPf fIPrPf	----	Δ Δ 0 -
LDX # <i>opr16i</i> LDX <i>opr8a</i> LDX <i>opr16a</i> LDX <i>opr0_xysp</i> LDX <i>opr9_xysp</i> LDX <i>opr16_xysp</i> LDX [D, <i>xysp</i>] LDX [<i>opr16_xysp</i>]	(M:M+1) ⇒ X Load Index Register X	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CE jj kk DE dd FE hh 11 EE xb EE xb ff EE xb ee ff EE xb EE xb ee ff	PO RPF RPO RPF RPO frPP fIFrPf fIPrPf	----	Δ Δ 0 -
LDY # <i>opr16i</i> LDY <i>opr8a</i> LDY <i>opr16a</i> LDY <i>opr0_xysp</i> LDY <i>opr9_xysp</i> LDY <i>opr16_xysp</i> LDY [D, <i>xysp</i>] LDY [<i>opr16_xysp</i>]	(M:M+1) ⇒ Y Load Index Register Y	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CD jj kk DD dd FD hh 11 ED xb ED xb ff ED xb ee ff ED xb ED xb ee ff	PO RPF RPO RPF RPO frPP fIFrPf fIPrPf	----	Δ Δ 0 -
LEAS <i>opr0_xysp</i> LEAS <i>opr9_xysp</i> LEAS <i>opr16_xysp</i>	Effective Address ⇒ SP Load Effective Address into SP	IDX IDX1 IDX2	1B xb 1B xb ff 1B xb ee ff	Pf PO PP	----	----
LEAX <i>opr0_xysp</i> LEAX <i>opr9_xysp</i> LEAX <i>opr16_xysp</i>	Effective Address ⇒ X Load Effective Address into X	IDX IDX1 IDX2	1A xb 1A xb ff 1A xb ee ff	Pf PO PP	----	----
LEAY <i>opr0_xysp</i> LEAY <i>opr9_xysp</i> LEAY <i>opr16_xysp</i>	Effective Address ⇒ Y Load Effective Address into Y	IDX IDX1 IDX2	19 xb 19 xb ff 19 xb ee ff	Pf PO PP	----	----
LSL <i>opr16a</i> LSL <i>opr0_xysp</i> LSL <i>opr9_xysp</i> LSL <i>opr16_xysp</i> LSL [D, <i>xysp</i>] LSL [<i>opr16_xysp</i>]	 Logical Shift Left same function as ASL	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	78 hh 11 68 xb 68 xb ff 68 xb ee ff 68 xb 68 xb ee ff	rPwO rPw rPwO frPPw fIFrPw fIPrPw	----	Δ Δ Δ Δ
LSLA LSLB	Logical Shift Accumulator A to Left Logical Shift Accumulator B to Left	INH INH	48 58	O O		
LSLD	 Logical Shift Left D Accumulator same function as ASLD	INH	59	O	----	Δ Δ Δ Δ
LSR <i>opr16a</i> LSR <i>opr0_xysp</i> LSR <i>opr9_xysp</i> LSR <i>opr16_xysp</i> LSR [D, <i>xysp</i>] LSR [<i>opr16_xysp</i>]	 Logical Shift Right	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	74 hh 11 64 xb 64 xb ff 64 xb ee ff 64 xb 64 xb ee ff	rPwO rPw rPwO frPPw fIFrPw fIPrPw	----	0 Δ Δ Δ
LSRA LSRB	Logical Shift Accumulator A to Right Logical Shift Accumulator B to Right	INH INH	44 54	O O		
LSRD	 Logical Shift Right D Accumulator	INH	49	O	----	0 Δ Δ Δ
MAXA <i>opr0_xysp</i> MAXA <i>opr9_xysp</i> MAXA <i>opr16_xysp</i> MAXA [D, <i>xysp</i>] MAXA [<i>opr16_xysp</i>]	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	----	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.

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
MAXM <i>opr0_xysp</i> MAXM <i>opr9_xysp</i> MAXM <i>opr16_xysp</i> MAXM [D, <i>xysp</i>] MAXM [<i>opr16_xysp</i>]	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 OfIfrPwP 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	--?–	????
MINA <i>opr0_xysp</i> MINA <i>opr9_xysp</i> MINA <i>opr16_xysp</i> MINA [D, <i>xysp</i>] MINA [<i>opr16_xysp</i>]	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 OfrrPP OfIfrPpf OfIPrPpf	----	Δ Δ Δ Δ
MINM <i>opr0_xysp</i> MINM <i>opr9_xysp</i> MINM <i>opr16_xysp</i> MINM [D, <i>xysp</i>] MINM [<i>opr16_xysp</i>]	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 OfrrPwP OfIfrPw OfIPrPw	----	Δ Δ Δ Δ
MOVB # <i>opr8</i> , <i>opr16a</i> ¹ MOVB # <i>opr8i</i> , <i>opr0_xysp</i> ¹ MOVB <i>opr16a</i> , <i>opr16a</i> ¹ MOVB <i>opr16a</i> , <i>opr0_xysp</i> ¹ MOVB <i>opr0_xysp</i> , <i>opr16a</i> ¹ MOVB <i>opr0_xysp</i> , <i>opr0_xysp</i> 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	----	----
MOVW # <i>opr16</i> , <i>opr16a</i> ¹ MOVW # <i>opr16i</i> , <i>opr0_xysp</i> ¹ MOVW <i>opr16a</i> , <i>opr16a</i> ¹ MOVW <i>opr16a</i> , <i>opr0_xysp</i> ¹ MOVW <i>opr0_xysp</i> , <i>opr16a</i> ¹ MOVW <i>opr0_xysp</i> , <i>opr0_xysp</i> ¹	(M:M+1 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	----	----
MUL	(A) × (B) ⇒ A:B 8 by 8 Unsigned Multiply	INH	12	O	----	---Δ
NEG <i>opr16a</i> NEG <i>opr0_xysp</i> NEG <i>opr9_xysp</i> NEG <i>opr16_xysp</i> NEG [D, <i>xysp</i>] NEG [<i>opr16_xysp</i>] NEGA NEGB	0 – (M) ⇒ M equivalent to (M̄) + 1 ⇒ M Two's Complement Negate 0 – (A) ⇒ A equivalent to (Ā) + 1 ⇒ A Negate Accumulator A 0 – (B) ⇒ B equivalent to (B̄) + 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	----	Δ Δ Δ Δ
NOP	No Operation	INH	A7	O	----	----
ORAA # <i>opr8i</i> ORAA <i>opr8a</i> ORAA <i>opr16a</i> ORAA <i>opr0_xysp</i> ORAA <i>opr9_xysp</i> ORAA <i>opr16_xysp</i> ORAA [D, <i>xysp</i>] ORAA [<i>opr16_xysp</i>]	(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 fIfrPpf fIPrPpf	----	Δ Δ 0 –

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

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
ORAB # <i>opr8i</i> ORAB <i>opr8a</i> ORAB <i>opr16a</i> ORAB <i>opr0_xysp</i> ORAB <i>opr9_xysp</i> ORAB <i>opr16_xysp</i> ORAB [D, <i>xysp</i>] ORAB [<i>opr16_xysp</i>]	(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 fIfrPpf fIPrPpf	----	Δ Δ 0 –
ORCC # <i>opr8i</i>	(CCR) + M ⇒ CCR Logical OR CCR with Memory	IMM	14 ii	P	↑ – ↑ ↑	↑ ↑ ↑ ↑ ↑
PSHA	(SP) – 1 ⇒ SP; (A) ⇒ M _(SP) Push Accumulator A onto Stack	INH	36	Os	----	----
PSHB	(SP) – 1 ⇒ SP; (B) ⇒ M _(SP) Push Accumulator B onto Stack	INH	37	Os	----	----
PSHC	(SP) – 1 ⇒ SP; (CCR) ⇒ M _(SP) Push CCR onto Stack	INH	39	Os	----	----
PSHD	(SP) – 2 ⇒ SP; (A:B) ⇒ M _(SP) :M _(SP+1) Push D Accumulator onto Stack	INH	3B	OS	----	----
PSHX	(SP) – 2 ⇒ SP; (X _H :X _L) ⇒ M _(SP) :M _(SP+1) Push Index Register X onto Stack	INH	34	OS	----	----
PSHY	(SP) – 2 ⇒ SP; (Y _H :Y _L) ⇒ M _(SP) :M _(SP+1) Push Index Register Y onto Stack	INH	35	OS	----	----
PULA	M _(SP) ⇒ A; (SP) + 1 ⇒ SP Pull Accumulator A from Stack	INH	32	ufo	----	----
PULB	M _(SP) ⇒ B; (SP) + 1 ⇒ SP Pull Accumulator B from Stack	INH	33	ufo	----	----
PULC	M _(SP) ⇒ CCR; (SP) + 1 ⇒ SP Pull CCR from Stack	INH	38	ufo	Δ ↓ Δ Δ	Δ Δ Δ Δ
PULD	M _(SP) :M _(SP+1) ⇒ A:B; (SP) + 2 ⇒ SP Pull D from Stack	INH	3A	Ufo	----	----
PULX	M _(SP) :M _(SP+1) ⇒ X _H :X _L ; (SP) + 2 ⇒ SP Pull Index Register X from Stack	INH	30	Ufo	----	----
PULY	M _(SP) :M _(SP+1) ⇒ Y _H :Y _L ; (SP) + 2 ⇒ SP Pull Index Register Y from Stack	INH	31	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 REWW. 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,	--?–	??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,RErf) (exit + re-entry replaces comma above if interrupted) ffff + Orf(t,	--?–	??Δ!

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C	Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
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	----	Δ Δ Δ Δ	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) ⇒ M Store Accumulator A to Memory	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5A dd 7A hh 11 6A xb 6A xb ff 6A xb 6A xb ee ff	Pw PwO Pw PwO PwP PIEW PIPw	----	Δ Δ 0 -
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	----	Δ Δ Δ Δ	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) ⇒ M Store Accumulator B to Memory	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5B dd 7B hh 11 6B xb 6B xb ff 6B xb 6B xb ee ff	Pw PwO Pw PwO PwP PIEW PIPw	----	Δ Δ 0 -
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	----	----	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) ⇒ M, (B) ⇒ M+1 Store Double Accumulator	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5C dd 7C hh 11 6C xb 6C xb ff 6C xb 6C xb ee ff	PW PWO PW PWO PWP PIEW PIPW	----	Δ Δ 0 -
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) uUUUUvPPP	Δ ↓ Δ Δ	Δ Δ Δ Δ	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) fvVEPPP (continue) ff (if STOP disabled) OO	----	----
RTS	$M_{(SP)}:M_{(SP+1)} \Rightarrow PC_H:PC_L;$ $(SP) + 2 \Rightarrow SP$ Return from Subroutine	INH	3D	UfPPP	----	----							
SBA	(A) - (B) ⇒ A Subtract B from A	INH	18 16	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 ⇒ 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	----	Δ Δ Δ Δ	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 11 6F xb 6F xb ff 6F xb ee ff 6F xb 6F xb ee ff	PW PWO PW PWO PWP PIEW PIPW	----	Δ Δ 0 -
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 ⇒ 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	----	Δ Δ Δ Δ	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 11 6E xb 6E xb ff 6E xb ee ff 6E xb 6E xb ee ff	PW PWO PW PWO PWP PIEW PIPW	----	Δ Δ 0 -
SEC	1 ⇒ C <i>Translates to ORCC #01</i>	IMM	14 01	P	----	--- 1	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 11 6D xb 6D xb ff 6D xb ee ff 6D xb 6D xb ee ff	PW PWO PW PWO PWP PIEW PIPW	----	Δ Δ 0 -
SEI	1 ⇒ I; (inhibit I interrupts) <i>Translates to ORCC #10</i>	IMM	14 10	P	--- 1	----							
SEV	1 ⇒ V <i>Translates to ORCC #02</i>	IMM	14 02	P	----	-- 1 -							
SEX <i>abc,dxys</i>	\$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 <i>Alternate mnemonic for TFR r1, r2</i>	INH	B7 eb	P	----	----	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) ⇒ A Subtract Memory from Accumulator A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	80 ii 90 dd B0 hh 11 A0 xb A0 xb ff A0 xb ee ff A0 xb A0 xb ee ff	P rPf rPO rPf rPO frPP fIFrPf fIPrPf	----	Δ Δ Δ Δ

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
SUBB #opr8i SUBB opr8a SUBB opr16a SUBB oprx0_xysp SUBB oprx9_xysp SUBB oprx16_xysp SUBB [D,xysp] SUBB [opr16,xysp]	(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	P rPf rPO rPf rPO frPP fIFrPf fIPrPf	----	Δ Δ Δ Δ
SUBD #opr16i SUBD opr8a SUBD opr16a SUBD oprx0_xysp SUBD oprx9_xysp SUBD oprx16_xysp SUBD [D,xysp] SUBD [opr16,xysp]	(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	----	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	---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	----	Δ Δ 0 -
TAP	(A) ⇒ CCR Translates to TFR A, CCR	INH	B7 02	P	Δ ↓ Δ Δ	Δ Δ Δ Δ
TBA	(B) ⇒ A Transfer B to A	INH	18 0F	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 1b rrr	PPP (branch) PPO (no branch)	----	----
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	----	Δ Δ - Δ ?
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 1b rrr	PPP (branch) PPO (no branch)	----	----
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	----	---- or Δ ↓ Δ Δ Δ Δ Δ Δ
TPA	(CCR) ⇒ A Translates to TFR CCR ,A	INH	B7 20	P	----	----

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12	S X H I	N Z V C
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	---1	----
TST opr16a TST oprx0_xysp TST oprx9_xysp TST oprx16_xysp TST [D,xysp] TST [opr16,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	----	Δ Δ 0 0
TSX	(SP) ⇒ X Translates to TFR SP,X	INH	B7 75	P	----	----
TSY	(SP) ⇒ Y Translates to TFR SP,Y	INH	B7 76	P	----	----
TXS	(X) ⇒ SP Translates to TFR X,SP	INH	B7 57	P	----	----
TYS	(Y) ⇒ SP Translates to TFR Y,SP	INH	B7 67	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	----	---- or ---1 ---- or -1-1 ----
WAV	$B \sum_{i=1}^B S_i F_i \Rightarrow Y:D \quad \text{and} \quad \sum_{i=1}^B F_i \Rightarrow X$ 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 (add if interrupt) SSS + UUUrr,	--?-	? Δ ? ?
wavr	see WAV	Special	3C	UUUrr,ffff (frr,ffff)O	--?-	? Δ ? ?
pseudo-instruction	Resume executing an interrupted WAV instruction (recover intermediate results from stack rather than initializing them to zero)			(exit + re-entry replaces comma above if interrupted) SSS + UUUrr,		
XGDY	(D) ⇔ (X) Translates to EXG D, X	INH	B7 C5	P	----	----
XGDY	(D) ⇔ (Y) Translates to EXG D, Y	INH	B7 C6	P	----	----