

Reference Guide

CPU12RG/D
Rev. 2, 11/2001

CPU12 Reference Guide
(for HCS12 and original
M68HC12)

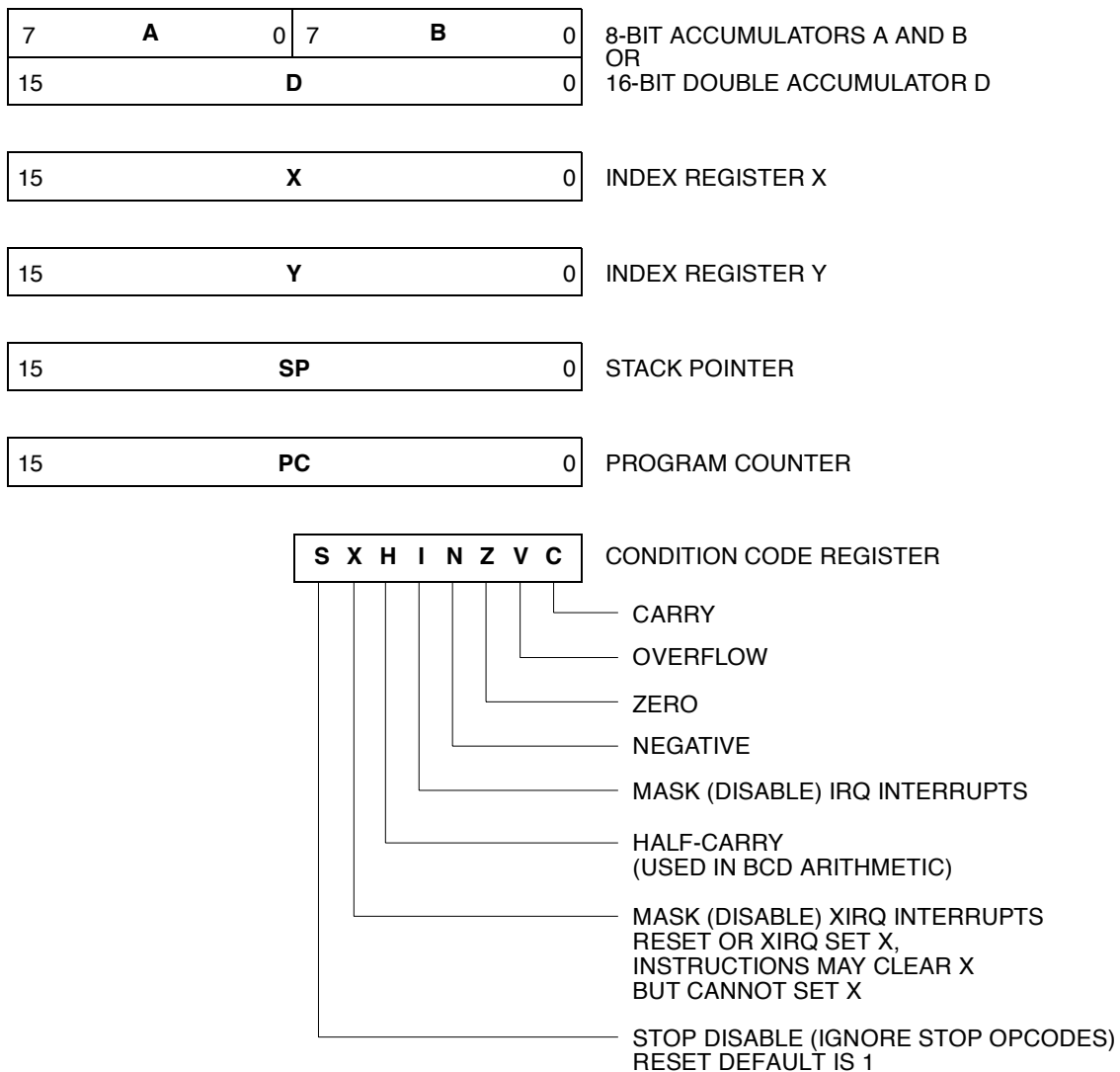
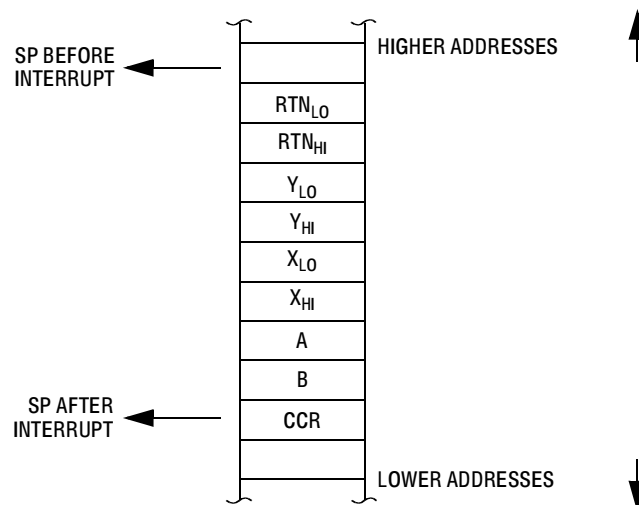


Figure 1. Programming Model

Stack and Memory Layout



STACK UPON ENTRY TO SERVICE ROUTINE
IF SP WAS ODD BEFORE INTERRUPT

SP +8	RTN _{LO}		SP +9
SP +6	Y _{LO}	RTN _{HI}	SP +7
SP +4	X _{LO}	Y _{HI}	SP +5
SP +2	A	X _{HI}	SP +3
SP	CCR	B	SP +1
SP -2			SP -1

STACK UPON ENTRY TO SERVICE ROUTINE
IF SP WAS EVEN BEFORE INTERRUPT

SP +9			SP +10
SP +7	RTN _{HI}	RTN _{LO}	SP +8
SP +5	Y _{HI}	Y _{LO}	SP +6
SP +4	X _{HI}	X _{LO}	SP +4
SP +1	B	A	SP +2
SP -1		CCR	SP

Interrupt Vector Locations

\$FFFE, \$FFFF	Power-On (POR) or External Reset
\$FFFC, \$FFFD	Clock Monitor Reset
\$FFFA, \$FFFB	Computer Operating Properly (COP Watchdog Reset)
\$FFF8, \$FFF9	Unimplemented Opcode Trap
\$FFF6, \$FFF7	Software Interrupt Instruction (SWI)
\$FFF4, \$FFF5	XIRQ
\$FFF2, \$FFF3	IRQ
\$FFC0-\$FFF1	Device-Specific Interrupt Sources

Notation Used in Instruction Set Summary

CPU Register Notation

Accumulator A — A or a
Accumulator B — B or b
Accumulator D — D or d
Index Register X — X or x

Index Register Y — Y or y
Stack Pointer — SP, sp, or s
Program Counter — PC, pc, or p
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
- opr0_xysp* — Indexed addressing postbyte code:
 - opr3,-xys* Predecrement X or Y or SP by 1 . . . 8
 - opr3,+xys* Preincrement X or Y or SP by 1 . . . 8
 - opr3,xys-* Postdecrement X or Y or SP by 1 . . . 8
 - opr3,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
- opr3* — Any positive integer 1 . . . 8 for pre/post increment/decrement
- opr5* — Any integer in the range -16 . . . +15
- opr9* — Any integer in the range -256 . . . +255
- opr16* — 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
 - $\overline{M}$ — Negation. One's complement (invert each bit of M)
 - :
- 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.

Continued on next page

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. See [Table 3](#) on page 22.
- 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. See [Table 4](#) on page 23.
- 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. See [Table 1](#) on page 20 and [Table 2](#) on page 21.

Access Detail

Each code letter except (,), and comma equals one CPU cycle.
 Uppercase = 16-bit operation and lowercase = 8-bit operation. For
 complex sequences see the *CPU12 Reference Manual*
 (CPU12RM/AD) for more detailed information.

- 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

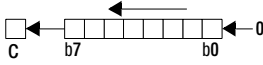
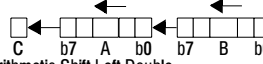
- — Status bit not affected by operation.
- 0 — Status bit cleared by operation.
- 1 — Status bit set by operation.
- Δ — Status bit affected by operation.
- ? — Status bit may be cleared or remain set, but is not set by operation.
- ↑ — Status bit may be set or remain cleared, but is not cleared by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

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	HC12		
ABA	(A) + (B) ⇒ A Add Accumulators A and B	INH	18 06	00	00	--Δ-	ΔΔΔΔ
ABX	(B) + (X) ⇒ X <i>Translates to LEAX B,X</i>	IDX	1A E5	Pf	PP ¹	----	----
ABY	(B) + (Y) ⇒ 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 [oprx16,xysp]	(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 A9 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP 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 [oprx16,xysp]	(B) + (M) + C ⇒ 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 E9 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP 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 [oprx16,xysp]	(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 AB xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP 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 [oprx16,xysp]	(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 EB xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP 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 [oprx16,xysp]	(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 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 [oprx16,xysp]	(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 A4 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP rPO frPP fIfrPf fIPrPf	----	ΔΔ0-
ANDB #opr8i ANDB opr8a ANDB opr16a ANDB oprx0_xysp ANDB oprx9_xysp ANDB oprx16_xysp ANDB [D,xysp] ANDB [oprx16,xysp]	(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 E4 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP rPO frPP fIfrPf fIPrPf	----	ΔΔ0-
ANDCC #opr8i	(CCR) • (M) ⇒ CCR Logical AND CCR with Memory	IMM	10 ii	P	P	↓↓↓↓	↓↓↓↓

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

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
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 rOPw rPw rPwO frPwP fIfPw fIPrPw O O	rOPw rPw rPwO 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 rPwO 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 rPPw frPPwOP	----	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 <i>CPU12 Reference Manual</i>	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 fIfP fIPrPf	P rPf rOP rPf rPO frPP fIfP 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 fIfP fIPrPf	P rPf rOP rPf rPO frPP fIfP 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.

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	HC12		
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	rPOw rPPw rPOw rPPw frPPwP	----	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.

Instruction Set Summary (Sheet 4 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12 HC12	S X H I	N Z V C
CMPA #opr8l CMPA opr8a CMPA opr16a CMPA oprx0_xysp CMPA oprx9_xysp CMPA oprx16_xysp CMPA [D,xysp] CMPA [opr16,xysp]	(A) – (M) Compare Accumulator A with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	81 ii 91 dd B1 hh 1l A1 xb A1 xb ff A1 xb ee ff A1 xb 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 #opr8l CMPB opr8a CMPB opr16a CMPB oprx0_xysp CMPB oprx9_xysp CMPB oprx16_xysp CMPB [D,xysp] CMPB [opr16,xysp]	(B) – (M) Compare Accumulator B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C1 ii D1 dd F1 hh 1l E1 xb E1 xb ff E1 xb ee ff E1 xb 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 [opr16,xysp] COMA COMB	$(\bar{M}) \Rightarrow M$ equivalent to $\$FF - (M) \Rightarrow M$ 1's Complement Memory Location $(\bar{A}) \Rightarrow A$ Complement Accumulator A $(\bar{B}) \Rightarrow B$ Complement Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	71 hh 1l 61 xb 61 xb ff 61 xb ee ff 61 xb 61 xb ee ff 41 51	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPwO frPPw fIfrPw fIPrPw O O	---- Δ Δ 0 1
CPD #opr16l CPD opr8a CPD opr16a CPD oprx0_xysp CPD oprx9_xysp CPD oprx16_xysp CPD [D,xysp] CPD [opr16,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 1l AC xb AC xb ff AC xb ee ff AC xb AC xb ee ff	PO rPf RPO rPf RPO frPP fIfRPf fIPRPf	OP rPf ROP rPf RPO frPP fIfrPf fIPrPf	---- Δ Δ Δ Δ
CPS #opr16l CPS opr8a CPS opr16a CPS oprx0_xysp CPS oprx9_xysp CPS oprx16_xysp CPS [D,xysp] CPS [opr16,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 1l AF xb AF xb ff AF xb ee ff AF xb AF xb ee ff	PO rPf RPO rPf RPO frPP fIfRPf fIPRPf	OP rPf ROP rPf RPO frPP fIfrPf fIPrPf	---- Δ Δ Δ Δ
CPX #opr16l CPX opr8a CPX opr16a CPX oprx0_xysp CPX oprx9_xysp CPX oprx16_xysp CPX [D,xysp] CPX [opr16,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 1l AE xb AE xb ff AE xb ee ff AE xb AE xb ee ff	PO rPf RPO rPf RPO frPP fIfRPf fIPRPf	OP rPf ROP rPf RPO frPP fIfrPf fIPrPf	---- Δ Δ Δ Δ
CPY #opr16l CPY opr8a CPY opr16a CPY oprx0_xysp CPY oprx9_xysp CPY oprx16_xysp CPY [D,xysp] CPY [opr16,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 1l AD xb AD xb ff AD xb ee ff AD xb AD xb ee ff	PO rPf RPO rPf RPO frPP fIfRPf fIPRPf	OP rPf ROP rPf 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	(cntr) – 1 $\Rightarrow$ 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 1b rr	PPP (branch) PPO (no branch)	PPP	---- ----

Instruction Set Summary (Sheet 5 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12 HC12	S X H I	N Z V C
DBNE <i>abdxys, rel9</i>	(cntr) – 1 $\Rightarrow$ cntr If (cntr) not = 0, then Branch; else Continue to next instruction Decrement Counter and Branch if $\neq$ 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 1b rr	PPP (branch) PPO (no branch)	PPPP	----
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 $\Rightarrow$ M Decrement Memory Location (A) – \$01 $\Rightarrow$ A Decrement A (B) – \$01 $\Rightarrow$ 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 63 xb ee ff 43 53	rPwO rPw rPwO frPwP fIfPrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfPrPw fIPrPw O O	---- $\Delta \Delta \Delta -$
DES	(SP) – \$0001 $\Rightarrow$ SP Translates to LEAS –1, SP	IDX	1B 9F	Pf	pp ¹	----
DEX	(X) – \$0001 $\Rightarrow$ X Decrement Index Register X	INH	09	O	O	---- - Δ --
DEY	(Y) – \$0001 $\Rightarrow$ Y Decrement Index Register Y	INH	03	O	O	---- - Δ --
EDIV	(Y:D) $\div$ (X) $\Rightarrow$ Y Remainder $\Rightarrow$ D 32 by 16 Bit $\Rightarrow$ 16 Bit Divide (unsigned)	INH	11	fffffffffffo	fffffffffffo	---- $\Delta \Delta \Delta \Delta$
EDIVS	(Y:D) $\div$ (X) $\Rightarrow$ Y Remainder $\Rightarrow$ D 32 by 16 Bit $\Rightarrow$ 16 Bit Divide (signed)	INH	18 14	Offfffffffffffo	Offfffffffffffo	---- $\Delta \Delta \Delta \Delta$
EMACS <i>opr16a</i> ²	(M _(X) :M _(X+1)) $\times$ (M _(Y) :M _(Y+1)) + (M – M + 3) $\Rightarrow$ M – M + 3 16 by 16 Bit $\Rightarrow$ 32 Bit Multiply and Accumulate (signed)	Special	18 12 hh 1l	ORROfffRRfWWP ORROfffRRfWWP		---- $\Delta \Delta \Delta \Delta$
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)) $\Rightarrow$ 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 OfRRPf OfIfRRPf OfIPRRPf	ORfP ORPO OfRRP OfIfRRP OfIPRRP	---- $\Delta \Delta \Delta \Delta$
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)) $\Rightarrow$ 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 OfRRPW OfIfRRPW OfIPRRPW	ORfP ORPWO OfRRP OfIfRRP OfIPRRP	---- $\Delta \Delta \Delta \Delta$
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)) $\Rightarrow$ 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 OfRRPf OfIfRRPf OfIPRRPf	ORfP ORPO OfRRP OfIfRRP OfIPRRP	---- $\Delta \Delta \Delta \Delta$
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)) $\Rightarrow$ 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 OfRRPW OfIfRRPW OfIPRRPW	ORfP ORPWO OfRRP OfIfRRP OfIPRRP	---- $\Delta \Delta \Delta \Delta$
EMUL	(D) $\times$ (Y) $\Rightarrow$ Y:D 16 by 16 Bit Multiply (unsigned)	INH	13	ffO	ffO	---- $\Delta \Delta - \Delta$
EMULS	(D) $\times$ (Y) $\Rightarrow$ Y:D 16 by 16 Bit Multiply (signed)	INH	18 13	OfO (if followed by page 2 instruction) OffO	OfO OfO	---- $\Delta \Delta - \Delta$
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) $\oplus$ (M) $\Rightarrow$ 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 fIfPrPf fIPrPf	P rPf rOP rPf rPO frPP fIfPrPf fIPrPf	---- $\Delta \Delta 0 -$

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.

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	HC12		
EORB #opr8l 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 ll E8 xb E8 xb ff E8 xb ee ff E8 xb E8 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rfP rOP rfP 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	----	$\Delta \Delta - \Delta ?$ 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	----	$-\Delta \Delta \Delta$
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	----	$-\Delta 0 \Delta$
IDIVS	$(D) \div (X) \Rightarrow X$; Remainder $\Rightarrow D$ 16 by 16 Bit Integer Divide (signed)	INH	18 15	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 ll 62 xb 62 xb ff 62 xb ee ff 62 xb 62 xb ee ff 42 52	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	$\Delta \Delta \Delta -$
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 ll 05 xb 05 xb ff 05 xb ee ff 05 xb 05 xb ee ff	PPP PPP PPP fPPP fIfPPP fIfPPP	PPP PPP PPP fPPP fIfPPP fIfPPP	----	----

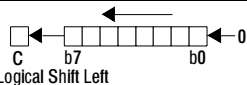
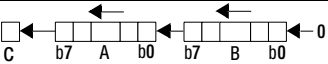
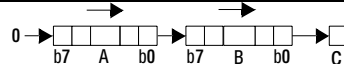
Note 1. Due to internal CPU requirements, the program word fetch is performed twice to the same address during this instruction.

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	HC12		
JSR <i>opr8a</i> JSR <i>opr16a</i> JSR <i>opr0_xysp</i> JSR <i>opr9_xysp</i> JSR <i>opr16_xysp</i> JSR [<i>D,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	----	----
LBLS <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 [<i>D,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 [<i>D,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 [<i>D,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 fIfRfP fIPRfP	----	Δ Δ 0 -

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

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	HC12		
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 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 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 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 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 64 xb ee ff 44 54	rPwO rPw rPwO frPPw 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 OfRPf OfIfRPf OfIPrPf	OrfP OrPO OfRPf OfIfRfP OfIPRfP	----	Δ Δ Δ Δ

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

Instruction Set Summary (Sheet 9 of 14)

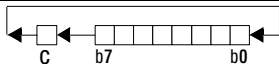
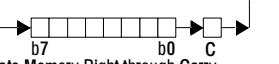
Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail		S X H I	N Z V C
				HCS12	HC12		
MAXM <i>opr</i> x0_ <i>xy</i> sp MAXM <i>opr</i> x9_ <i>xy</i> sp MAXM <i>opr</i> x16_ <i>xy</i> sp MAXM [D, <i>xy</i> sp] MAXM [<i>opr</i> x16_ <i>xy</i> sp]	MAX((A), (M)) ⇒ M MAX of 2 Unsigned 8-Bit 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 OfIfrPwP OfIfrPw OfIPrPw	OrPw OrPwO OfIfrPwP OfIfrPw OfIPrPw	----	Δ Δ Δ Δ
MEM	μ (grade) ⇒ $M_{\gamma\gamma}$; (X) + 4 ⇒ X; (Y) + 1 ⇒ Y; A unchanged if (A) < P1 or (A) > P2 then $\mu = 0$, else $\mu = \text{MIN}(((A) - P1) \times S1, (P2 - (A)) \times S2, \$FF)$ 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 OfIfrPwP OfIfrPw OfIPrPw	OrfP OrPO OfIfrPwP OfIfrPw OfIPrPw	----	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 OfIfrPwP OfIfrPw OfIPrPw	OrPw OrPwO OfIfrPwP 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> 16, <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 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 $(\bar{M}) + 1 \Rightarrow M$ Two's Complement Negate 0 – (A) ⇒ A equivalent to $(\bar{A}) + 1 \Rightarrow A$ Negate Accumulator A 0 – (B) ⇒ B equivalent to $(\bar{B}) + 1 \Rightarrow B$ Negate Accumulator B	EXT IDX IDX1 IDX2 [D, IDX] [IDX2] INH INH	70 hh ll 60 xb 60 xb ff 60 xb ee ff 60 xb 60 xb ee ff 40 50	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfrPw fIPrPw O O	----	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 fIfrPw fIPrPw	P rfP rOP rfP rPO frPP fIfrPw fIPrPw	----	Δ Δ 0 –

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

Instruction Set Summary (Sheet 10 of 14)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail	HCS12	HC12	S X H I	N Z V C
ORAB #opr8i ORAB opr8a ORAB opr16a ORAB oprx0_xysp ORAB oprx9_xysp ORAB oprx16_xysp ORAB [D,xysp] ORAB [opr16,xysp]	(B) + (M) ⇒ B Logical OR B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CA ii DA dd FA hh ll EA xb EA xb ff EA xb ee ff EA xb EA xb ee ff	P rPf rPO rPf rPO frPP fIfrrPf fIPrrPf	P rPf rOP rPf rPO frPP fIfrrPf fIPrrPf	P rPf rOP rPf rPO frPP fIfrrPf fIPrrPf	----	Δ Δ 0 -
ORCC #opr8i	(CCR) + M ⇒ CCR Logical OR CCR with Memory	IMM	14 ii	P	P	P	↑ ↑ ↑ ↑	↑ ↑ ↑ ↑
PSHA	(SP) - 1 ⇒ SP; (A) ⇒ M _(SP) Push Accumulator A onto Stack	INH	36	Os	Os	Os	----	----
PSHB	(SP) - 1 ⇒ SP; (B) ⇒ M _(SP) Push Accumulator B onto Stack	INH	37	Os	Os	Os	----	----
PSHC	(SP) - 1 ⇒ SP; (CCR) ⇒ M _(SP) Push CCR onto Stack	INH	39	Os	Os	Os	----	----
PSHD	(SP) - 2 ⇒ SP; (A:B) ⇒ M _(SP) :M _(SP+1) Push D Accumulator onto Stack	INH	3B	OS	OS	OS	----	----
PSHX	(SP) - 2 ⇒ SP; (X _H :X _L) ⇒ M _(SP) :M _(SP+1) Push Index Register X onto Stack	INH	34	OS	OS	OS	----	----
PSHY	(SP) - 2 ⇒ SP; (Y _H :Y _L) ⇒ M _(SP) :M _(SP+1) Push Index Register Y onto Stack	INH	35	OS	OS	OS	----	----
PULA	(M _(SP)) ⇒ A; (SP) + 1 ⇒ SP Pull Accumulator A from Stack	INH	32	ufo	ufo	ufo	----	----
PULB	(M _(SP)) ⇒ B; (SP) + 1 ⇒ SP Pull Accumulator B from Stack	INH	33	ufo	ufo	ufo	----	----
PULC	(M _(SP)) ⇒ CCR; (SP) + 1 ⇒ SP Pull CCR from Stack	INH	38	ufo	ufo	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	Ufo	Ufo	----	----
PULX	(M _(SP) :M _(SP+1)) ⇒ X _H :X _L ; (SP) + 2 ⇒ SP Pull Index Register X from Stack	INH	30	Ufo	Ufo	Ufo	----	----
PULY	(M _(SP) :M _(SP+1)) ⇒ Y _H :Y _L ; (SP) + 2 ⇒ SP Pull Index Register Y from Stack	INH	31	Ufo	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, ff + Orf(t,	Orf(t,tx)O (exit + re-entry replaces comma above if interrupted) ff + Orf(t, 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 \$FFFF 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) (r,RfRf) (exit + re-entry replaces comma above if interrupted) fff + ORf(t, fff + ORf(t,	ORf(t,Tx)O (loop to read weight if enabled) (r,RfRf) (r,RfRf) (exit + re-entry replaces comma above if interrupted) fff + ORf(t, fff + ORf(t,	ORf(t,Tx)O (loop to read weight if enabled) (r,RfRf) (r,RfRf) (exit + re-entry replaces comma above if interrupted) fff + ORf(t, fff + ORf(t,	-- ? -	? ? Δ !

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	HC12		
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 fIfPrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfPrPw fIPrPw O O	---- 	Δ Δ 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 fIfPrPw fIPrPw O O	rOPw rPw rPOw frPPw fIfPrPw fIPrPw O O	---- 	Δ Δ 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
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 rPf rOP rPf rPO frPP fIfrPf fIPrPf	---- 	Δ Δ 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 rPf rOP rPf rPO frPP fIfrPf fIPrPf	---- 	Δ Δ 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	----	----

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	HC12		
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 1l 6A xb 6A xb ff 6A xb ee ff 6A xb 6A xb ee ff	Pw PwO Pw PwO PwP PIfPw PIPPw	Pw wOP Pw PwO PwP PIfPw PIPPw	----	Δ Δ 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) ⇒ 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 PIfPw PIPPw	Pw wOP Pw PwO PwP PIfPw PIPPw	----	Δ Δ 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) ⇒ M, (B) ⇒ 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 PIfPW PIPPW	PW WOP PW PWO PWP PIfPW PIPPW	----	Δ Δ 0 -
STOP	(SP) - 2 ⇒ SP; RTN _H :RTN _L ⇒ M _(SP) :M _(SP+1) ; (SP) - 2 ⇒ SP; (Y _H :Y _L) ⇒ M _(SP) :M _(SP+1) ; (SP) - 2 ⇒ SP; (X _H :X _L) ⇒ M _(SP) :M _(SP+1) ; (SP) - 2 ⇒ SP; (B:A) ⇒ M _(SP) :M _(SP+1) ; (SP) - 1 ⇒ SP; (CCR) ⇒ M _(SP) ; STOP All Clocks Registers stacked to allow quicker recovery by interrupt. If S control bit = 1, the STOP instruction is disabled and acts like a two-cycle NOP.	INH	18 3E	(entering STOP) OOSSSSsf OOSSSfSs (exiting STOP) fVfPPP fVfPPP (continue) ff fO (if STOP disabled) OO 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) ⇒ 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 PIfPW PIPPW	PW WOP PW PWO PWP PIfPW PIPPW	----	Δ Δ 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) ⇒ 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 PIfPW PIPPW	PW WOP PW PWO PWP PIfPW PIPPW	----	Δ Δ 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) ⇒ 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 PIfPW PIPPW	PW WOP PW PWO PWP PIfPW PIPPW	----	Δ Δ 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) ⇒ 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	----	Δ Δ Δ Δ

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	HC12	
SUBB #opr8l 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 E0 xb ee ff	P rPf rPO rPf rPO frPP fIfRPf fIPrPf	P rPf rOP rPf rPO frPP fIfRPf fIPrPf	---- Δ Δ Δ Δ
SUBD #opr16l 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	OP RfP ROP RfP 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	VSPSSPSsP* VfPPP	---- 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 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 rr	PPP (branch) PPO (no branch)	PPP	---- ----
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	ORffffP OrrffffP	OrffffP	---- Δ Δ – Δ ? C Bit is undefined in HC12
TBNE abdxys,rel9	If (cntr) not = 0, then Branch; else Continue to next instruction Test Counter and Branch if Not Zero (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 1b rr	PPP (branch) PPO (no branch)	PPP	---- ----
TFR abcdxys,abcdxys	(r1) ⇒ r2 or \$00:(r1) ⇒ r2 or (r1[7:0]) ⇒ r2 Transfer Register to Register r1 and r2 may be A, B, CCR, D, X, Y, or SP	INH	B7 eb	P	P	---- or Δ ↓ Δ Δ Δ Δ Δ Δ
TPA	(CCR) ⇒ A Translates to TFR CCR, A	INH	B7 20	P	P	---- ----

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	HC12		
TRAP <i>trapnum</i>	(SP) − 2 ⇒ SP; RTN _H :RTN _L ⇒ M _{(SP);M_(SP+1)} ; (SP) − 2 ⇒ SP; (Y _H :Y _L) ⇒ M _{(SP);M_(SP+1)} ; (SP) − 2 ⇒ SP; (X _H :X _L) ⇒ M _{(SP);M_(SP+1)} ; (SP) − 2 ⇒ SP; (B:A) ⇒ M _{(SP);M_(SP+1)} ; (SP) − 1 ⇒ SP; (CCR) ⇒ M _(SP) ; 1 ⇒ I; (TRAP Vector) ⇒ PC Unimplemented opcode trap	INH	18 tn tn = \$30−\$39 or \$40−\$FF	OVSPSSPSsP	OFVSPSSPSsP	---1	----
TST <i>opr16a</i> TST <i>oprX0_xysp</i> TST <i>oprX9_xysp</i> TST <i>oprX16_xysp</i> TST [D, <i>xysp</i>] TST [<i>oprX16_xysp</i>] TSTA TSTB	(M) − 0 Test Memory for Zero or Minus (A) − 0 Test A for Zero or Minus (B) − 0 Test B for Zero or Minus	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	F7 hh ll E7 xb E7 xb ff E7 xb ee ff E7 xb E7 xb ee ff 97 D7	rPO rPf rPO frPP fIfrPf fIPrPf O O	rOP rFP rPO frPP fIfrFP fIPrFP O O	----	Δ Δ 0 0
TSX	(SP) ⇒ X <i>Translates to</i> TFR SP,X	INH	B7 75	P	P	----	----
TSY	(SP) ⇒ Y <i>Translates to</i> TFR SP,Y	INH	B7 76	P	P	----	----
TXS	(X) ⇒ SP <i>Translates to</i> TFR X,SP	INH	B7 57	P	P	----	----
TYS	(Y) ⇒ SP <i>Translates to</i> TFR Y,SP	INH	B7 67	P	P	----	----
WAI	(SP) − 2 ⇒ SP; RTN _H :RTN _L ⇒ M _{(SP);M_(SP+1)} ; (SP) − 2 ⇒ SP; (Y _H :Y _L) ⇒ M _{(SP);M_(SP+1)} ; (SP) − 2 ⇒ SP; (X _H :X _L) ⇒ M _{(SP);M_(SP+1)} ; (SP) − 2 ⇒ SP; (B:A) ⇒ M _{(SP);M_(SP+1)} ; (SP) − 1 ⇒ SP; (CCR) ⇒ M _(SP) ; WAIT for interrupt	INH	3E	OSSSSsf	OSSSfSsf	----	----
				(after interrupt)			or
				fVfPPP	VfPPP	----	----
							or
						-1-1	----
WAV	B $\sum_{i=1} S_i F_i \Rightarrow Y:D$ and $\sum_{i=1} F_i \Rightarrow X$ Calculate Sum of Products and Sum of Weights for Weighted Average Calculation Initialize B, X, and Y before WAV. B specifies number of elements. X points at first element in S _i list. Y points at first element in F _i list. All S _i and F _i elements are 8-bits. If interrupted, six extra bytes of stack used for intermediate values	Special	18 3C	Of(frr,ffff)O Off(frr,fffff)O (add if interrupt) SSS + UUUrr, SSSf + UUUrr		--?-	? Δ ??
wavr	see WAV	Special	3C	UUUrr,ffff	UUUrrfffff	--?-	? Δ ??
pseudo-instruction	Resume executing an interrupted WAV instruction (recover intermediate results from stack rather than initializing them to zero)			(frr,ffff)O (exit + re-entry replaces comma above if interrupted)	(frr,fffff)O		
				SSS + UUUrr, SSSf + UUUrr			
XGDY	(D) ⇔ (X) <i>Translates to</i> EXG D, X	INH	B7 C5	P	P	----	----
XGDY	(D) ⇔ (Y) <i>Translates to</i> EXG D, Y	INH	B7 C6	P	P	----	----

Table 1. Indexed Addressing Mode Postbyte Encoding (xb)

00	0,X 5b const	10	-16,X 5b const	20	1,X+ pre-inc	30	1,X+ post-inc	40	0,Y 5b const	50	-16,Y 5b const	60	1,Y+ pre-inc	70	1,Y+ post-inc	80	0,SP 5b const	90	-16,SP 5b const	A0	1,SP+ pre-inc	B0	1,SP+ post-inc	C0	0,PC 5b const	D0	-16,PC 5b const	E0	n,X 9b const	F0	n,SP 9b const
01	1,X 5b const	11	-15,X 5b const	21	2,X+ pre-inc	31	2,X+ post-inc	41	1,Y 5b const	51	-15,Y 5b const	61	2,Y+ pre-inc	71	2,Y+ post-inc	81	1,SP 5b const	91	-15,SP 5b const	A1	2,SP+ pre-inc	B1	2,SP+ post-inc	C1	1,PC 5b const	D1	-15,PC 5b const	E1	-n,X 9b const	F1	-n,SP 9b const
02	2,X 5b const	12	-14,X 5b const	22	3,X+ pre-inc	32	3,X+ post-inc	42	2,Y 5b const	52	-14,Y 5b const	62	3,Y+ pre-inc	72	3,Y+ post-inc	82	2,SP 5b const	92	-14,SP 5b const	A2	3,SP+ pre-inc	B2	3,SP+ post-inc	C2	2,PC 5b const	D2	-14,PC 5b const	E2	n,X 16b const	F2	n,SP 16b const
03	3,X 5b const	13	-13,X 5b const	23	4,X+ pre-inc	33	4,X+ post-inc	43	3,Y 5b const	53	-13,Y 5b const	63	4,Y+ pre-inc	73	4,Y+ post-inc	83	3,SP 5b const	93	-13,SP 5b const	A3	4,SP+ pre-inc	B3	4,SP+ post-inc	C3	3,PC 5b const	D3	-13,PC 5b const	E3	[n,X] 16b indir	F3	[n,SP] 16b indir
04	4,X 5b const	14	-12,X 5b const	24	5,X+ pre-inc	34	5,X+ post-inc	44	4,Y 5b const	54	-12,Y 5b const	64	5,Y+ pre-inc	74	5,Y+ post-inc	84	4,SP 5b const	94	-12,SP 5b const	A4	5,SP+ pre-inc	B4	5,SP+ post-inc	C4	4,PC 5b const	D4	-12,PC 5b const	E4	A,X A offset	F4	A,SP A offset
05	5,X 5b const	15	-11,X 5b const	25	6,X+ pre-inc	35	6,X+ post-inc	45	5,Y 5b const	55	-11,Y 5b const	65	6,Y+ pre-inc	75	6,Y+ post-inc	85	5,SP 5b const	95	-11,SP 5b const	A5	6,SP+ pre-inc	B5	6,SP+ post-inc	C5	5,PC 5b const	D5	-11,PC 5b const	E5	B,X B offset	F5	B,SP B offset
06	6,X 5b const	16	-10,X 5b const	26	7,X+ pre-inc	36	7,X+ post-inc	46	6,Y 5b const	56	-10,Y 5b const	66	7,Y+ pre-inc	76	7,Y+ post-inc	86	6,SP 5b const	96	-10,SP 5b const	A6	7,SP+ pre-inc	B6	7,SP+ post-inc	C6	6,PC 5b const	D6	-10,PC 5b const	E6	D,X D offset	F6	D,SP D offset
07	7,X 5b const	17	-9,X 5b const	27	8,X+ pre-inc	37	8,X+ post-inc	47	7,Y 5b const	57	-9,Y 5b const	67	8,Y+ pre-inc	77	8,Y+ post-inc	87	7,SP 5b const	97	-9,SP 5b const	A7	8,SP+ pre-inc	B7	8,SP+ post-inc	C7	7,PC 5b const	D7	-9,PC 5b const	E7	[D,X] D indirect	F7	[D,SP] D indirect
08	8,X 5b const	18	-8,X 5b const	28	8,X- pre-inc	38	8,X- post-inc	48	8,Y 5b const	58	-8,Y 5b const	68	8,Y- pre-inc	78	8,Y- post-inc	88	8,SP 5b const	98	-8,SP 5b const	A8	8,SP- pre-inc	B8	8,SP- post-inc	C8	8,PC 5b const	D8	-8,PC 5b const	E8	n,Y 9b const	F8	n,PC 9b const
09	9,X 5b const	19	-7,X 5b const	29	7,X- pre-inc	39	7,X- post-inc	49	9,Y 5b const	59	-7,Y 5b const	69	7,Y- pre-inc	79	7,Y- post-inc	89	9,SP 5b const	99	-7,SP 5b const	A9	7,SP- pre-inc	B9	7,SP- post-inc	C9	9,PC 5b const	D9	-7,PC 5b const	E9	-n,Y 9b const	F9	-n,PC 9b const
0A	10,X 5b const	1A	-6,X 5b const	2A	6,X- pre-inc	3A	6,X- post-inc	4A	10,Y 5b const	5A	-6,Y 5b const	6A	6,Y- pre-inc	7A	6,Y- post-inc	8A	10,SP 5b const	9A	-6,SP 5b const	AA	6,SP- pre-inc	BA	6,SP- post-inc	CA	10,PC 5b const	DA	-6,PC 5b const	EA	n,Y 16b const	FA	n,PC 16b const
0B	11,X 5b const	1B	-5,X 5b const	2B	5,X- pre-inc	3B	5,X- post-inc	4B	11,Y 5b const	5B	-5,Y 5b const	6B	5,Y- pre-inc	7B	5,Y- post-inc	8B	11,SP 5b const	9B	-5,SP 5b const	AB	5,SP- pre-inc	BB	5,SP- post-inc	CB	11,PC 5b const	DB	-5,PC 5b const	EB	[n,Y] 16b indir	FB	[n,PC] 16b indir
0C	12,X 5b const	1C	-4,X 5b const	2C	4,X- pre-inc	3C	4,X- post-inc	4C	12,Y 5b const	5C	-4,Y 5b const	6C	4,Y- pre-inc	7C	4,Y- post-inc	8C	12,SP 5b const	9C	-4,SP 5b const	AC	4,SP- pre-inc	BC	4,SP- post-inc	CC	12,PC 5b const	DC	-4,PC 5b const	EC	A,Y A offset	FC	A,PC A offset
0D	13,X 5b const	1D	-3,X 5b const	2D	3,X- pre-inc	3D	3,X- post-inc	4D	13,Y 5b const	5D	-3,Y 5b const	6D	3,Y- pre-inc	7D	3,Y- post-inc	8D	13,SP 5b const	9D	-3,SP 5b const	AD	3,SP- pre-inc	BD	3,SP- post-inc	CD	13,PC 5b const	DD	-3,PC 5b const	ED	B,Y B offset	FD	B,PC B offset
0E	14,X 5b const	1E	-2,X 5b const	2E	2,X- pre-inc	3E	2,X- post-inc	4E	14,Y 5b const	5E	-2,Y 5b const	6E	2,Y- pre-inc	7E	2,Y- post-inc	8E	14,SP 5b const	9E	-2,SP 5b const	AE	2,SP- pre-inc	BE	2,SP- post-inc	CE	14,PC 5b const	DE	-2,PC 5b const	EE	D,Y D offset	FE	D,PC D offset
0F	15,X 5b const	1F	-1,X 5b const	2F	1,X- pre-inc	3F	1,X- post-inc	4F	15,Y 5b const	5F	-1,Y 5b const	6F	1,Y- pre-inc	7F	1,Y- post-inc	8F	15,SP 5b const	9F	-1,SP 5b const	AF	1,SP- pre-inc	BF	1,SP- post-inc	CF	15,PC 5b const	DF	-1,PC 5b const	EF	[D,Y] D indirect	FF	[D,PC] D indirect

Key to Table 1

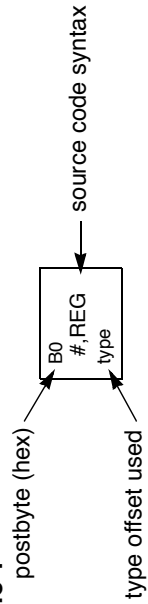


Table 2. Indexed Addressing Mode Summary

Postbyte Code (xb)	Operand Syntax	Comments
rr0nnnnn	,r n,r -n,r	5-bit constant offset n = -16 to +15 rr can specify X, Y, SP, or PC
111rr0zs	n,r -n,r	Constant offset (9- or 16-bit signed) z- 0 = 9-bit with sign in LSB of postbyte (s) 1 = 16-bit if z = s = 1, 16-bit offset indexed-indirect (see below) rr can specify X, Y, SP, or PC
rr1pnnnn	n,-r n,+r n,r- n,r+	Auto predecrement, preincrement, postdecrement, or postincrement; p = pre-(0) or post-(1), n = -8 to -1, +1 to +8 rr can specify X, Y, or SP (PC not a valid choice)
111rr1aa	A,r B,r D,r	Accumulator offset (unsigned 8-bit or 16-bit) aa - 00 = A 01 = B 10 = D (16-bit) 11 = see accumulator D offset indexed-indirect rr can specify X, Y, SP, or PC
111rr011	[n,r]	16-bit offset indexed-indirect rr can specify X, Y, SP, or PC
111rr111	[D,r]	Accumulator D offset indexed-indirect rr can specify X, Y, SP, or PC

Table 3. Transfer and Exchange Postbyte Encoding

TRANSFERS									
↓ LS	MS⇒	0	1	2	3	4	5	6	7
0		A ⇒ A	B ⇒ A	CCR ⇒ A	TMP3 _L ⇒ A	B ⇒ A	X _L ⇒ A	Y _L ⇒ A	SP _L ⇒ A
1		A ⇒ B	B ⇒ B	CCR ⇒ B	TMP3 _L ⇒ B	B ⇒ B	X _L ⇒ B	Y _L ⇒ B	SP _L ⇒ B
2		A ⇒ CCR	B ⇒ CCR	CCR ⇒ CCR	TMP3 _L ⇒ CCR	B ⇒ CCR	X _L ⇒ CCR	Y _L ⇒ CCR	SP _L ⇒ CCR
3		sex:A ⇒ TMP2	sex:B ⇒ TMP2	sex:CCR ⇒ TMP2	TMP3 ⇒ TMP2	D ⇒ TMP2	X ⇒ TMP2	Y ⇒ TMP2	SP ⇒ TMP2
4		sex:A ⇒ D SEX A,D	sex:B ⇒ D SEX B,D	sex:CCR ⇒ D SEX CCR,D	TMP3 ⇒ D	D ⇒ D	X ⇒ D	Y ⇒ D	SP ⇒ D
5		sex:A ⇒ X SEX A,X	sex:B ⇒ X SEX B,X	sex:CCR ⇒ X SEX CCR,X	TMP3 ⇒ X	D ⇒ X	X ⇒ X	Y ⇒ X	SP ⇒ X
6		sex:A ⇒ Y SEX A,Y	sex:B ⇒ Y SEX B,Y	sex:CCR ⇒ Y SEX CCR,Y	TMP3 ⇒ Y	D ⇒ Y	X ⇒ Y	Y ⇒ Y	SP ⇒ Y
7		sex:A ⇒ SP SEX A,SP	sex:B ⇒ SP SEX B,SP	sex:CCR ⇒ SP SEX CCR,SP	TMP3 ⇒ SP	D ⇒ SP	X ⇒ SP	Y ⇒ SP	SP ⇒ SP
EXCHANGES									
↓ LS	MS⇒	8	9	A	B	C	D	E	F
0		A ⇔ A	B ⇔ A	CCR ⇔ A	TMP3 _L ⇔ A \$00:A ⇒ TMP3	B ⇒ A A ⇒ B	X _L ⇔ A \$00:A ⇒ X	Y _L ⇔ A \$00:A ⇒ Y	SP _L ⇔ A \$00:A ⇒ SP
1		A ⇔ B	B ⇔ B	CCR ⇔ B	TMP3 _L ⇔ B \$FF:B ⇒ TMP3	B ⇒ B \$FF ⇒ A	X _L ⇔ B \$FF:B ⇒ X	Y _L ⇔ B \$FF:B ⇒ Y	SP _L ⇔ B \$FF:B ⇒ SP
2		A ⇔ CCR	B ⇔ CCR	CCR ⇔ CCR	TMP3 _L ⇔ CCR \$FF:CCR ⇒ TMP3	B ⇒ CCR \$FF:CCR ⇒ D	X _L ⇔ CCR \$FF:CCR ⇒ X	Y _L ⇔ CCR \$FF:CCR ⇒ Y	SP _L ⇔ CCR \$FF:CCR ⇒ SP
3		\$00:A ⇒ TMP2 TMP2 _L ⇒ A	\$00:B ⇒ TMP2 TMP2 _L ⇒ B	\$00:CCR ⇒ TMP2 TMP2 _L ⇒ CCR	TMP3 ⇔ TMP2	D ⇔ TMP2	X ⇔ TMP2	Y ⇔ TMP2	SP ⇔ TMP2
4		\$00:A ⇒ D	\$00:B ⇒ D	\$00:CCR ⇒ D B ⇒ CCR	TMP3 ⇔ D	D ⇔ D	X ⇔ D	Y ⇔ D	SP ⇔ D
5		\$00:A ⇒ X X _L ⇒ A	\$00:B ⇒ X X _L ⇒ B	\$00:CCR ⇒ X X _L ⇒ CCR	TMP3 ⇔ X	D ⇔ X	X ⇔ X	Y ⇔ X	SP ⇔ X
6		\$00:A ⇒ Y Y _L ⇒ A	\$00:B ⇒ Y Y _L ⇒ B	\$00:CCR ⇒ Y Y _L ⇒ CCR	TMP3 ⇔ Y	D ⇔ Y	X ⇔ Y	Y ⇔ Y	SP ⇔ Y
7		\$00:A ⇒ SP SP _L ⇒ A	\$00:B ⇒ SP SP _L ⇒ B	\$00:CCR ⇒ SP SP _L ⇒ CCR	TMP3 ⇔ SP	D ⇔ SP	X ⇔ SP	Y ⇔ SP	SP ⇔ SP

TMP2 and TMP3 registers are for factory use only.

Table 4. Loop Primitive Postbyte Encoding (1b)

00 DBEQ (+)	A	10 DBEQ (-)	B	20 DBNE (+)	A	30 DBNE (-)	A	40 TBEQ (+)	A	50 TBEQ (-)	A	60 TBNE (+)	A	70 TBNE (-)	A	80 IBEQ (+)	A	90 IBEQ (-)	A	A0 IBNE (+)	A	B0 IBNE (-)	A
01 DBEQ (+)	B	11 DBEQ (-)	B	21 DBNE (+)	B	31 DBNE (-)	B	41 TBEQ (+)	B	51 TBEQ (-)	B	61 TBNE (+)	B	71 TBNE (-)	B	81 IBEQ (+)	B	91 IBEQ (-)	B	A1 IBNE (+)	B	B1 IBNE (-)	B
02 —		12 —		22 —		32 —		42 —		52 —		62 —		72 —		82 —		92 —		A2 —		B2 —	
03 —		13 —		23 —		33 —		43 —		53 —		63 —		73 —		83 —		93 —		A3 —		B3 —	
04 DBEQ (+)	D	14 DBEQ (-)	D	24 DBNE (+)	D	34 DBNE (-)	D	44 TBEQ (+)	D	54 TBEQ (-)	D	64 TBNE (+)	D	74 TBNE (-)	D	84 IBEQ (+)	D	94 IBEQ (-)	D	A4 IBNE (+)	D	B4 IBNE (-)	D
05 DBEQ (+)	X	15 DBEQ (-)	X	25 DBNE (+)	X	35 DBNE (-)	X	45 TBEQ (+)	X	55 TBEQ (-)	X	65 TBNE (+)	X	75 TBNE (-)	X	85 IBEQ (+)	X	95 IBEQ (-)	X	A5 IBNE (+)	X	B5 IBNE (-)	X
06 DBEQ (+)	Y	16 DBEQ (-)	Y	26 DBNE (+)	Y	36 DBNE (-)	Y	46 TBEQ (+)	Y	56 TBEQ (-)	Y	66 TBNE (+)	Y	76 TBNE (-)	Y	86 IBEQ (+)	Y	96 IBEQ (-)	Y	A6 IBNE (+)	Y	B6 IBNE (-)	Y
07 DBEQ (+)	SP	17 DBEQ (-)	SP	27 DBNE (+)	SP	37 DBNE (-)	SP	47 TBEQ (+)	SP	57 TBEQ (-)	SP	67 TBNE (+)	SP	77 TBNE (-)	SP	87 IBEQ (+)	SP	97 IBEQ (-)	SP	A7 IBNE (+)	SP	B7 IBNE (-)	SP

Key to Table 4

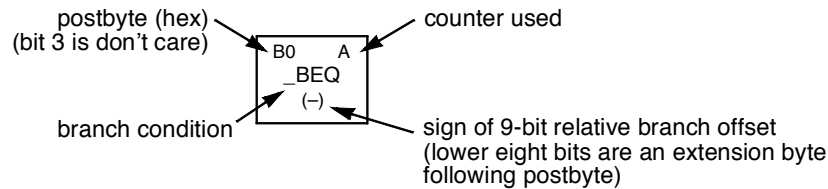


Table 5. Branch/Complementary Branch

Branch				Complementary Branch			
Test	Mnemonic	Opcode	Boolean	Test	Mnemonic	Opcode	Comment
$r > m$	BGT	2E	$Z + (N \oplus V) = 0$	$r \leq m$	BLE	2F	Signed
$r \geq m$	BGE	2C	$N \oplus V = 0$	$r < m$	BLT	2D	Signed
$r = m$	BEQ	27	$Z = 1$	$r \neq m$	BNE	26	Signed
$r \leq m$	BLE	2F	$Z + (N \oplus V) = 1$	$r > m$	BGT	2E	Signed
$r < m$	BLT	2D	$N \oplus V = 1$	$r \geq m$	BGE	2C	Signed
$r > m$	BHI	22	$C + Z = 0$	$r \leq m$	BLS	23	Unsigned
$r \geq m$	BHS/BCC	24	$C = 0$	$r < m$	BLO/BCS	25	Unsigned
$r = m$	BEQ	27	$Z = 1$	$r \neq m$	BNE	26	Unsigned
$r \leq m$	BLS	23	$C + Z = 1$	$r > m$	BHI	22	Unsigned
$r < m$	BLO/BCS	25	$C = 1$	$r \geq m$	BHS/BCC	24	Unsigned
Carry	BCS	25	$C = 1$	No Carry	BCC	24	Simple
Negative	BMI	2B	$N = 1$	Plus	BPL	2A	Simple
Overflow	BVS	29	$V = 1$	No Overflow	BVC	28	Simple
$r = 0$	BEQ	27	$Z = 1$	$r \neq 0$	BNE	26	Simple
Always	BRA	20	—	Never	BRN	21	Unconditional

For 16-bit offset long branches precede opcode with a \$18 page prebyte.

Memory Expansion

There are three basic memory expansion configurations in the M68HC12 and HCS12 MCU Families.

1. Basic 64 Kbyte memory map with no additional expanded memory support
2. >5 megabyte expanded memory support with 8-bit PPAGE, DPAGE, and EPAGE registers (MC68HC812A4 only)
3. >1 megabyte expanded memory support with 6-bit PPAGE register only — This configuration applies to all currently available HC12 and HCS12 devices with >60 Kbytes of on-chip FLASH memory.

Memory precedence

- Highest —
- On-chip registers (usually \$0000 or \$1000)
- BDM ROM (only when BDM active)
- On-chip RAM
- On-chip EEPROM
- On-chip program memory (FLASH or ROM)
- Expansion windows (on MCUs with expanded memory)
- Other external memory
- Lowest —

CPU sees 64 Kbytes of address space (CPU_ADDR [15:0])

- PPAGE 8-bit register to select 1 of 256 — 16 Kbyte program pages
- or 6-bit register to select 1 of 64 — 16 Kbyte program pages
- DPAGE 8-bit register to select 1 of 256 — 4 Kbyte data pages
- EPAGE 8-bit register to select 1 of 256 — 1 Kbyte extra pages

Extended address is up to 22 bits (EXT_ADDR [21:0])

Program expansion window works with CALL and RTC instructions to simplify program access to extended memory space. Data and extra expansion windows (when present) use traditional banked expansion memory techniques.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

Freescale Semiconductor, Inc.

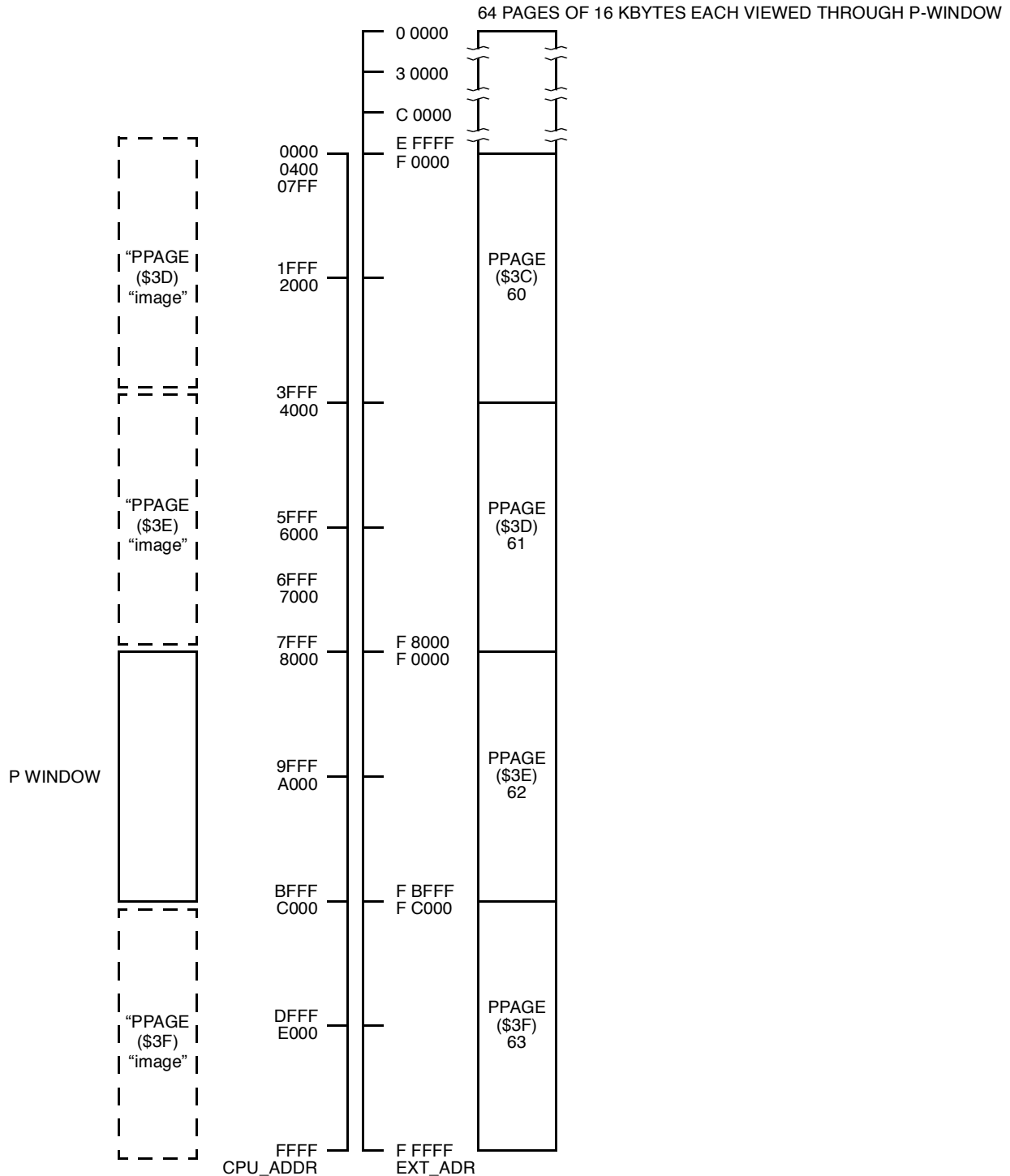


Figure 2. Memory Mapping in 1-Megabyte Map

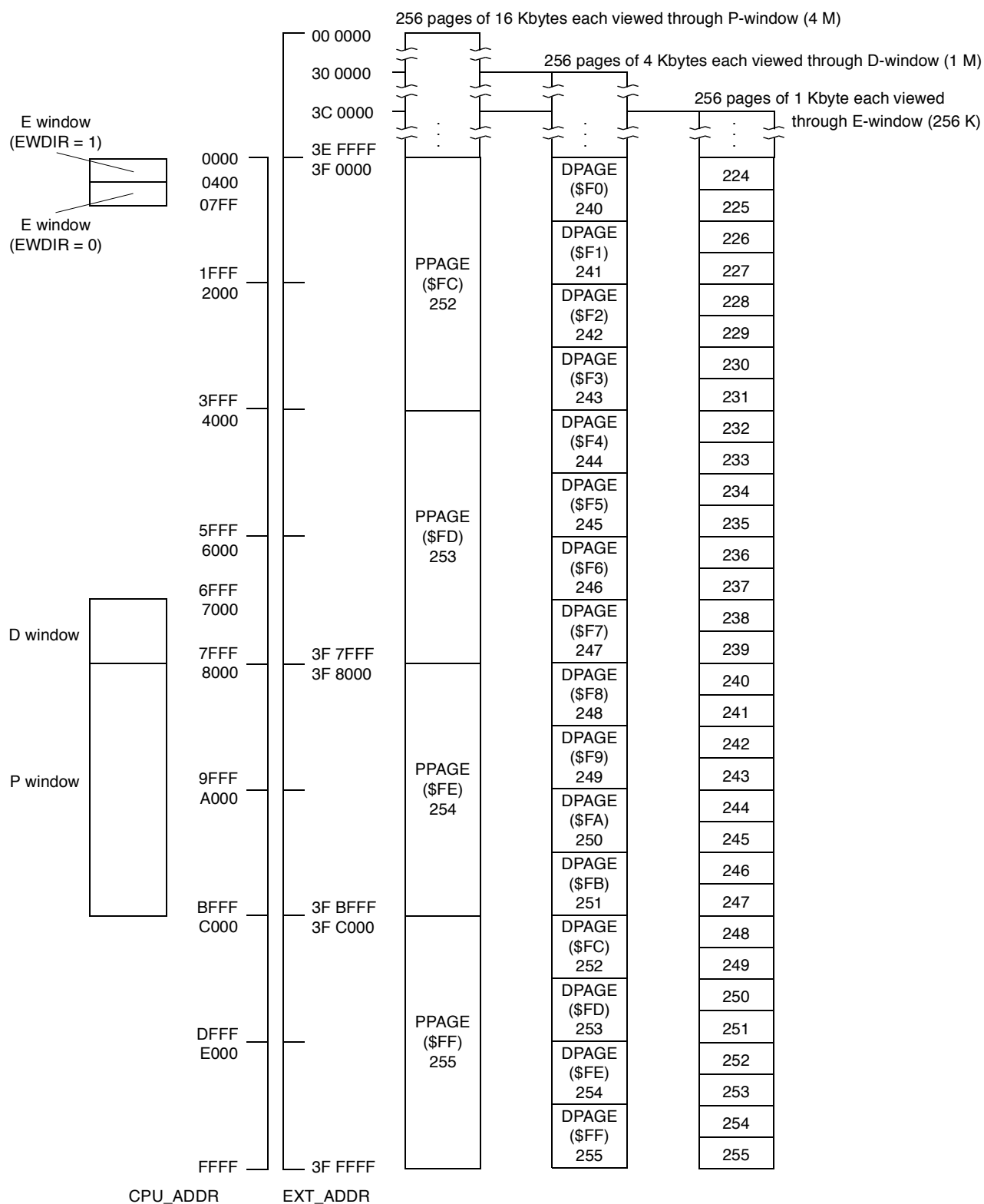


Figure 3. Memory Mapping in 4-Megabyte Map

Table 6. CPU12 Opcode Map (Sheet 1 of 2)

00	BGND	ANDCC	BRA	PULX	NEGA	NEGB	NEG	3-6	70	4	80	SUBA	90	3	A0	3-6	B0	3	C0	1	D0	3	E0	3-6	F0	3	
01	MEM	EDIV	BRN	PULY	COMA	COMB	COM	3-6	71	4	81	CMPA	91	3	A1	3-6	B1	3	C1	1	D1	3	E1	3-6	F1	3	
02	INY	MUL	BHI	PULA	INCA	INCB	INC	3-6	72	4	82	SBCA	92	3	A2	3-6	B2	3	C2	1	D2	3	E2	3-6	F2	3	
03	DEY	EMUL	BLS	PULB	DECA	DECB	DEC	3-6	73	4	83	SUBD	93	3	A3	3-6	B3	3	C3	2	D3	3	E3	3-6	F3	3	
04	loop	ORCC	BCC	PSHX	LSRA	LSRB	LSR	3-6	74	4	84	ANDA	94	3	A4	3-6	B4	3	C4	1	D4	3	E4	3-6	F4	3	
05	3-6	15	4-7	25	3/1	35	2	65	3-6	75	4	85	BITA	95	3	A5	3-6	B5	3	C5	1	D5	3	E5	3-6	F5	3
06	JMP	JSR	BCS	PSHY	ROLA	ROLB	ROL	3-6	76	4	86	LDAA	96	3	A6	3-6	B6	3	C6	1	D6	3	E6	3-6	F6	3	
07	JMP	JSR	BNE	PSHA	RORA	RORB	ROR	3-6	77	4	87	CLRA	97	1	A7	1	B7	1	C7	1	D7	1	E7	3-6	F7	3	
08	INX	page 2	BVC	PULC	ASLA	ASLB	ASL	3-6	78	4	88	EORA	98	3	A8	3-6	B8	3	C8	1	D8	3	E8	3-6	F8	3	
09	DEX	LEAY	BVS	PSHC	LSRD	ASLD	CLR	3-6	79	3	89	ADCA	99	3	A9	3-6	B9	3	C9	1	D9	3	E9	3-6	F9	3	
0A	RTC	LEAX	BPL	PULD	CALL	STAA	STAB	3-6	7A	3	8A	ORAA	9A	3	AA	3-6	BA	3	CA	1	DA	3	EA	3-6	FA	3	
0B	RTI	LEAS	BMI	PSHD	CALL	STAB	STAB	3-6	7B	3	8B	ADDA	9B	3	AB	3-6	BB	3	CB	1	DB	3	EB	3-6	FB	3	
0C	BSET	BSET	BGE	3-6	15	4-7	25	6C	3-6	7C	3	8C	CPD	9C	3	AC	3-6	BC	3	CC	2	DC	3	EC	3-6	FC	3
0D	BCLR	BCLR	BLT	RTS	BCLR	STY	STY	3-6	7D	3	8D	CPY	9D	3	AD	3-6	BD	3	CD	2	DD	3	ED	3-6	FD	3	
0E	BRSET	BRSET	BGT	WAI	BRSET	STX	STX	3-6	7E	3	8E	CPX	9E	3	AE	3-6	BE	3	CE	2	DE	3	EE	3-6	FE	3	
0F	BRCLR	BRCLR	BLE	SWI	BRCLR	STS	STS	3-6	7F	3	8F	CPS	9F	3	AF	3-6	BF	3	CF	2	DF	3	EF	3-6	FF	3	
10	BRCLR	BRCLR	BLE	SWI	BRCLR	STS	STS	3-6	7F	3	8F	CPS	9F	3	AF	3-6	BF	3	CF	2	DF	3	EF	3-6	FF	3	

Key to Table 6:

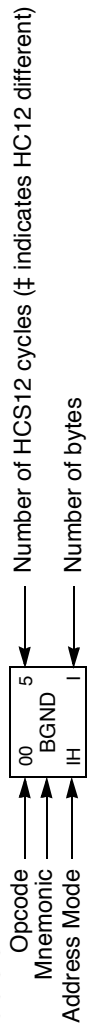


Table 6. CPU12 Opcode Map (Sheet 2 of 2)

00	MOVW	4	10	10	12	20	4	30	10	40	10	50	10	60	10	70	10	80	10	90	10	A0	10	B0	10	C0	10	D0	10	E0	10	F0	10	
	IM-ID	5	IH	2	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
01	MOVW	5	11	12	21	3	31	10	41	10	51	10	61	10	71	10	81	10	91	10	A1	10	B1	10	C1	10	D1	10	E1	10	F1	10	1	
	EX-ID	5	IH	2	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
02	MOVW	5	12	13	22	4/3	32	10	42	10	52	10	62	10	72	10	82	10	92	10	A2	10	B2	10	C2	10	D2	10	E2	10	F2	10	2	
	ID-ID	4	SP	4	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
03	MOVW	5	13	3	23	4/3	33	10	43	10	53	10	63	10	73	10	83	10	93	10	A3	10	B3	10	C3	10	D3	10	E3	10	F3	10	3	
	IM-EX	6	IH	2	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
04	MOVW	6	14	12	24	4/3	34	10	44	10	54	10	64	10	74	10	84	10	94	10	A4	10	B4	10	C4	10	D4	10	E4	10	F4	10	4	
	EX-EX	6	IH	2	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
05	MOVW	5	15	12	25	4/3	35	10	45	10	55	10	65	10	75	10	85	10	95	10	A5	10	B5	10	C5	10	D5	10	E5	10	F5	10	5	
	ID-EX	5	IH	2	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
06	ABA	2	16	2	26	4/3	36	10	46	10	56	10	66	10	76	10	86	10	96	10	A6	10	B6	10	C6	10	D6	10	E6	10	F6	10	6	
	IH	2	IH	2	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
07	DAA	3	17	2	27	4/3	37	10	47	10	57	10	67	10	77	10	87	10	97	10	A7	10	B7	10	C7	10	D7	10	E7	10	F7	10	7	
	IH	2	IH	2	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
08	MOVW	4	18	4-7	28	4/3	38	10	48	10	58	10	68	10	78	10	88	10	98	10	A8	10	B8	10	C8	10	D8	10	E8	10	F8	10	8	
	IM-ID	4	ID	3-5	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
09	MOVW	5	19	4-7	29	4/3	39	10	49	10	59	10	69	10	79	10	89	10	99	10	A9	10	B9	10	C9	10	D9	10	E9	10	F9	10	9	
	EX-ID	5	ID	3-5	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
0A	MOVW	5	1A	4-7	2A	4/3	3A	10	4A	10	5A	10	6A	10	7A	10	8A	10	9A	10	AA	10	BA	10	CA	10	DA	10	EA	10	FA	10	10	
	ID-ID	4	ID	3-5	RL	4	SP	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
0B	MOVW	4	1B	4-7	2B	4/3	3B	10	4B	10	5B	10	6B	10	7B	10	8B	10	9B	10	AB	10	BB	10	CB	10	DB	10	EB	10	FB	10	11	
	IM-EX	5	ID	3-5	RL	4	SP	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
0C	MOVW	6	1C	4-7	2C	4/3	3C	10	4C	10	5C	10	6C	10	7C	10	8C	10	9C	10	AC	10	BC	10	CC	10	DC	10	EC	10	FC	10	12	
	EX-EX	6	ID	3-5	RL	4	SP	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
0D	MOVW	5	1	D4-7	2D	4/3	3D	10	4D	10	5D	10	6D	10	7D	10	8D	10	9D	10	AD	10	BD	10	CD	10	DD	10	ED	10	FD	10	13	
	ID-EX	5	ID	3-5	RL	4	ID	3	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
0E	TAB	2	1E	4-7	2E	4/3	3E	10	4E	10	5E	10	6E	10	7E	10	8E	10	9E	10	AE	10	BE	10	CE	10	DE	10	EE	10	FE	10	14	
	IH	2	ID	3-5	RL	4	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2
0F	TBA	2	1F	4-7	2F	4/3	3F	10	4F	10	5F	10	6F	10	7F	10	8F	10	9F	10	AF	10	BF	10	CF	10	DF	10	EF	10	FF	10	15	
	ID	2	ID	3-5	RL	4	ID	3	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2	IH	2

* The opcode \$04 (on sheet 1 of 2) corresponds to one of the loop primitive instructions DBEQ, DBNE, IBNE, IBEQ, TBNE, or TBNE.

† Refer to instruction summary for more information.

‡ Refer to instruction summary for different HC12 cycle count.

Table 7. Hexadecimal to ASCII Conversion

Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII
\$00	NUL	\$20	SP <i>space</i>	\$40	@	\$60	' <i>grave</i>
\$01	SOH	\$21	!	\$41	A	\$61	a
\$02	STX	\$22	" <i>quote</i>	\$42	B	\$62	b
\$03	ETX	\$23	#	\$43	C	\$63	c
\$04	EOT	\$24	\$	\$44	D	\$64	d
\$05	ENQ	\$25	%	\$45	E	\$65	e
\$06	ACK	\$26	&	\$46	F	\$66	f
\$07	BEL <i>beep</i>	\$27	' <i>apost.</i>	\$47	G	\$67	g
\$08	BS <i>back sp</i>	\$28	(	\$48	H	\$68	h
\$09	HT <i>tab</i>	\$29	)	\$49	I	\$69	i
\$0A	LF <i>linefeed</i>	\$2A	*	\$4A	J	\$6A	j
\$0B	VT	\$2B	+	\$4B	K	\$6B	k
\$0C	FF	\$2C	, <i>comma</i>	\$4C	L	\$6C	l
\$0D	CR <i>return</i>	\$2D	- <i>dash</i>	\$4D	M	\$6D	m
\$0E	SO	\$2E	. <i>period</i>	\$4E	N	\$6E	n
\$0F	SI	\$2F	/	\$4F	O	\$6F	o
\$10	DLE	\$30	0	\$50	P	\$70	p
\$11	DC1	\$31	1	\$51	Q	\$71	q
\$12	DC2	\$32	2	\$52	R	\$72	r
\$13	DC3	\$33	3	\$53	S	\$73	s
\$14	DC4	\$34	4	\$54	T	\$74	t
\$15	NAK	\$35	5	\$55	U	\$75	u
\$16	SYN	\$36	6	\$56	V	\$76	v
\$17	ETB	\$37	7	\$57	W	\$77	w
\$18	CAN	\$38	8	\$58	X	\$78	x
\$19	EM	\$39	9	\$59	Y	\$79	y
\$1A	SUB	\$3A	:	\$5A	Z	\$7A	z
\$1B	ESCAPE	\$3B	;	\$5B	[	\$7B	{
\$1C	FS	\$3C	<	\$5C	\	\$7C	
\$1D	GS	\$3D	=	\$5D	]	\$7D	}
\$1E	RS	\$3E	>	\$5E	^	\$7E	~
\$1F	US	\$3F	?	\$5F	_ <i>under</i>	\$7F	DEL <i>delete</i>

Hexadecimal to Decimal Conversion

To convert a hexadecimal number (up to four hexadecimal digits) to decimal, look up the decimal equivalent of each hexadecimal digit in [Table 8](#). The decimal equivalent of the original hexadecimal number is the sum of the weights found in the table for all hexadecimal digits.

Table 8. Hexadecimal to/from Decimal Conversion

15 Bit 8				7 Bit 0			
15 12		11	8	7 4		3	0
4th Hex Digit		3rd Hex Digit		2nd Hex Digit		1st Hex Digit	
Hex	Decimal	Hex	Decimal	Hex	Decimal	Hex	Decimal
0	0	0	0	0	0	0	0
1	4,096	1	256	1	16	1	1
2	8,192	2	512	2	32	2	2
3	12,288	3	768	3	48	3	3
4	16,384	4	1,024	4	64	4	4
5	20,480	5	1,280	5	80	5	5
6	24,576	6	1,536	6	96	6	6
7	28,672	7	1,792	7	112	7	7
8	32,768	8	2,048	8	128	8	8
9	36,864	9	2,304	9	144	9	9
A	40,960	A	2,560	A	160	A	10
B	45,056	B	2,816	B	176	B	11
C	49,152	C	3,072	C	192	C	12
D	53,248	D	3,328	D	208	D	13
E	57,344	E	3,484	E	224	E	14
F	61,440	F	3,840	F	240	F	15

Decimal to Hexadecimal Conversion

To convert a decimal number (up to 65,535₁₀) to hexadecimal, find the largest decimal number in [Table 8](#) that is less than or equal to the number you are converting. The corresponding hexadecimal digit is the most significant hexadecimal digit of the result. Subtract the decimal number found from the original decimal number to get the *remaining decimal value*. Repeat the procedure using the remaining decimal value for each subsequent hexadecimal digit.

HOW TO REACH US:

USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution;
P.O. Box 5405, Denver, Colorado 80217
1-303-675-2140 or 1-800-441-2447

JAPAN:

Motorola Japan Ltd.; SPS, Technical Information Center,
3-20-1, Minami-Azabu Minato-ku, Tokyo 106-8573 Japan
81-3-3440-3569

ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.;
Silicon Harbour Centre, 2 Dai King Street,
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong
852-26668334

TECHNICAL INFORMATION CENTER:

1-800-521-6274

HOME PAGE:

<http://www.motorola.com/semiconductors>

Information in this document is provided solely to enable system and software implementers to use Motorola products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Motorola data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.



Motorola and the Stylized M Logo are registered in the U.S. Patent and Trademark Office. digital dna is a trademark of Motorola, Inc. All other product or service names are the property of their respective owners. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© Motorola, Inc. 2001

CPU12RG/D

**For More Information On This Product,
Go to: www.freescale.com**