Appendix A
Instruction Reference

A.1 Introduction

This appendix provides quick references for the instruction set, opcode map, and encoding.

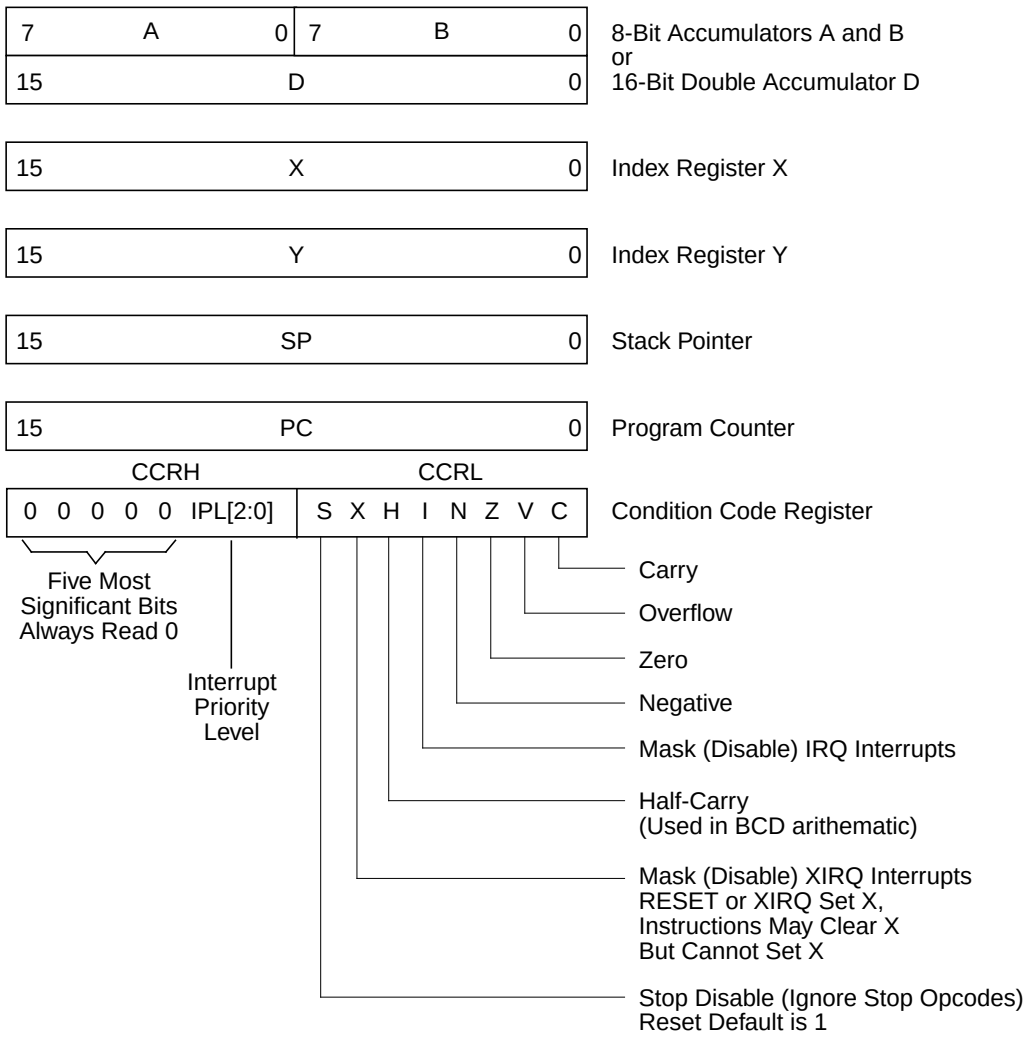
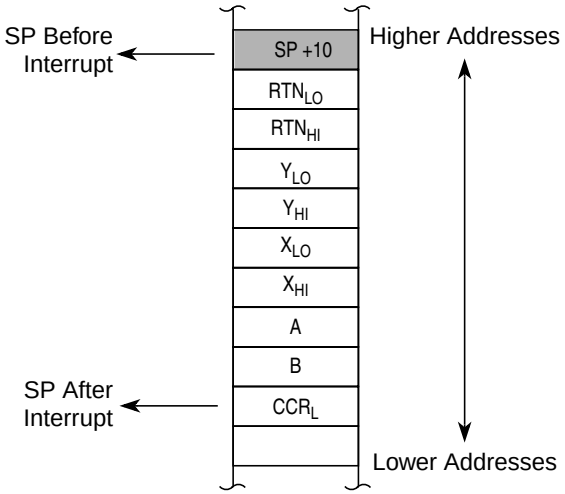


Figure A-1. Programming Model

A.2 Stack and Memory Layout

HCS12



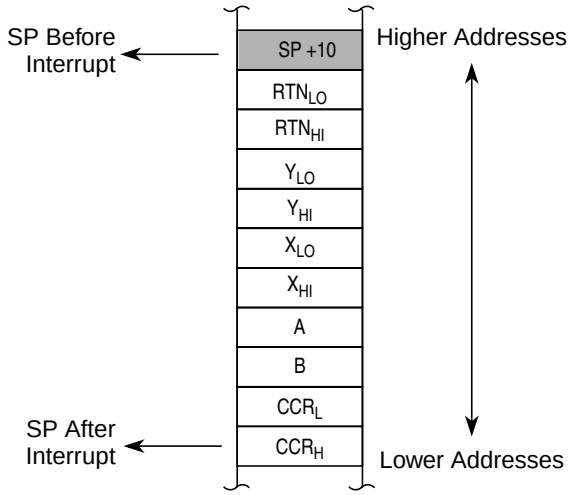
STACK UPON ENTRY TO SERVICE ROUTINE
IF SP WAS ODD BEFORE INTERRUPT

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

HCS12X



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

SP +9	RTN_{LO}		SP +10
SP +7	Y_{LO}	RTN_{HI}	SP +8
SP +5	X_{LO}	Y_{HI}	SP +6
SP +3	A	X_{HI}	SP +4
SP +1	CCR_L	B	SP +2
SP –1		CCR_H	SP

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

SP +10			SP +11
SP +8	RTN_{HI}	RTN_{LO}	SP +9
SP +6	Y_{HI}	Y_{LO}	SP +7
SP +4	X_{HI}	X_{LO}	SP +5
SP +2	B	A	SP +3
SP	CCR_H	CCR_L	SP +1

A.3 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
$FF00–$FFF1	Device-Specific Interrupt Sources

A.4 Notation Used in Instruction Set Summary

CPU12 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_xys — 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
oprx5,*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 –128 to +127 locations
rel9 — Label of branch destination within –256 to +255 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)
: — 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.
⇒ — 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 A-5.

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 A-6.

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 A-3 and Table A-4.

Access Detail

Each code letter except (,), and comma equals one CPU12 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, CPU12 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

NA — Not available

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.

Table A-1. Instruction Set Summary (Sheet 1 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
ABA	(A) + (B) ⇒ A Add Accumulators A and B	INH	18 06	OO	OO	--Δ-	Δ Δ Δ Δ
ABX	(B) + (X) ⇒ X *Translates to* LEAX B,X	IDX	1A E5	Pf	Pf	----	----
ABY	(B) + (Y) ⇒ Y *Translates to* LEAY B,Y	IDX	19 ED	Pf	Pf	----	----
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 ll A9 xb A9 xb ff A9 xb ee ff A9 xb A9 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	--Δ-	Δ Δ Δ Δ
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 ll E9 xb E9 xb ff E9 xb ee ff E9 xb E9 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	--Δ-	Δ Δ Δ Δ

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 ll
			AB xb
			AB xb ff
			AB xb ee ff
			AB xb
			AB xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rPf
					rPO
					rPf
					rPO
					frPP
					fIfrPf
					fIPrPf
						--Δ-	Δ Δ Δ Δ

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 ll
			EB xb
			EB xb ff
			EB xb ee ff
			EB xb
			EB xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rPf
					rPO
					rPf
					rPO
					frPP
					fIfrPf
					fIPrPf
						--Δ-	Δ Δ Δ Δ

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 ll
			E3 xb
			E3 xb ff
			E3 xb ee ff
			E3 xb
			E3 xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					PO
					RPf
					RPO
					RPf
					RPO
					fRPP
					fIfRPf
					fIPRPf
						----	Δ Δ Δ Δ

ADDX #opr16i **ADDX opr8a** **ADDX opr16a** **ADDX oprx0_xysp** **ADDX oprx9,xysp** **ADDX oprx16,xysp** **ADDX [D,xysp]** **ADDX [oprx16,xysp]**	**(X) + (M:M+1) ⇒ X** **Add without Carry to X**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 8B jj kk** **18 9B dd** **18 BB hh ll** **18 AB xb** **18 AB xb ff** **18 AB xb ee ff** **18 AB xb** **18 AB xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	--Δ-	Δ Δ Δ Δ
ADDY #opr16i **ADDY opr8a** **ADDY opr16a** **ADDY oprx0_xysp** **ADDY oprx9,xysp** **ADDY oprx16,xysp** **ADDY [D,xysp]** **ADDY [oprx16,xysp]**	**(Y) + (M:M+1) ⇒ Y** **Add without Carry to Y**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 CB jj kk** **18 DB dd** **18 FB hh ll** **18 EB xb** **18 EB xb ff** **18 EB xb ee ff** **18 EB xb** **18 EB xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	--Δ-	Δ Δ Δ Δ
ADED #opr16i **ADED opr8a** **ADED opr16a** **ADED oprx0_xysp** **ADED oprx9,xysp** **ADED oprx16,xysp** **ADED [D,xysp]** **ADED [oprx16,xysp]**	**(A:B) + (M:M+1) + C ⇒ A:B** **Add with Carry to D (A:B)**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 C3 jj kk** **18 D3 dd** **18 F3 hh ll** **18 E3 xb** **18 E3 xb ff** **18 E3 xb ee ff** **18 E3 xb** **18 E3 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	--Δ-	Δ Δ Δ Δ

Table A-1. Instruction Set Summary (Sheet 2 of 20)

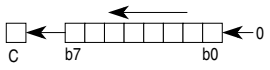
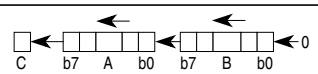
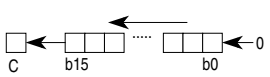
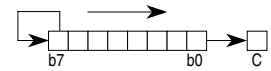
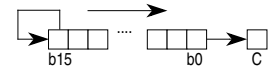
Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
ADEX #opr16i **ADEX opr8a** **ADEX opr16a** **ADEX oprx0_xysp** **ADEX oprx9,xysp** **ADEX oprx16,xysp** **ADEX [D,xysp]** **ADEX [oprx16,xysp]**	**(X) + (M:M+1) + C ⇒ X** **Add with Carry to X**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 89 jj kk** **18 99 dd** **18 B9 hh ll** **18 A9 xb** **18 A9 xb ff** **18 A9 xb ee ff** **18 A9 xb** **18 A9 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	– – Δ –	Δ Δ Δ Δ
ADEY #opr16i **ADEY opr8a** **ADEY opr16a** **ADEY oprx0_xysp** **ADEY oprx9,xysp** **ADEY oprx16,xysp** **ADEY [D,xysp]** **ADEY [oprx16,xysp]**	**(Y) + (M:M+1) + C ⇒ Y** **Add with Carry to Y**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 C9 jj kk** **18 D9 dd** **18 F9 hh ll** **18 E9 xb** **18 E9 xb ff** **18 E9 xb ee ff** **18 E9 xb** **18 E9 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	– – Δ –	Δ Δ Δ Δ
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 ll A4 xb A4 xb ff A4 xb ee ff A4 xb A4 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	– – – –	Δ Δ 0 –
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 ll E4 xb E4 xb ff E4 xb ee ff E4 xb E4 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	– – – –	Δ Δ 0 –
ANDCC #opr8i	(CCR) • (M) ⇒ CCR Logical AND CCR with Memory	IMM	10 ii	P	P	⇓ ⇓ ⇓ ⇓	⇓ ⇓ ⇓ ⇓
ANDX #opr16i **ANDX opr8a** **ANDX opr16a** **ANDX oprx0_xysp** **ANDX oprx9,xysp** **ANDX oprx16,xysp** **ANDX [D,xysp]** **ANDX [oprx16,xysp]**	**(X) • (M:M+1) ⇒ X** **Logical AND X with Memory**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 84 jj kk** **18 94 dd** **18 B4 hh ll** **18 A4 xb** **18 A4 xb ff** **18 A4 xb ee ff** **18 A4 xb** **18 A4 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	– – – –	Δ Δ 0 –
ANDY #opr16i **ANDY opr8a** **ANDY opr16a** **ANDY oprx0_xysp** **ANDY oprx9,xysp** **ANDY oprx16,xysp** **ANDY [D,xysp]** **ANDY [oprx16,xysp]**	**(Y) • (M:M+1) ⇒ Y** **Logical AND Y with Memory**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 C4 jj kk** **18 D4 dd** **18 F4 hh ll** **18 E4 xb** **18 E4 xb ff** **18 E4 xb ee ff** **18 E4 xb** **18 E4 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	– – – –	Δ Δ 0 –
ASL opr16a ASL oprx0_xysp ASL oprx9,xysp ASL oprx16,xysp ASL [D,xysp] ASL [oprx16,xysp] ASLA ASLB	 Arithmetic Shift Left Arithmetic Shift Left Accumulator A Arithmetic Shift Left Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	78 hh ll 68 xb 68 xb ff 68 xb ee ff 68 xb 68 xb ee ff 48 58	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	– – – –	Δ Δ Δ Δ
ASLD	 Arithmetic Shift Left Double	INH	59	O	O	– – – –	Δ Δ Δ Δ
ASLW opr16a **ASLW oprx0_xysp** **ASLW oprx9,xysp** **ASLW oprx16,xysp** **ASLW [D,xysp]** **ASLW [oprx16,xysp]** **ASLX** **ASLY**	 **Arithmetic Shift Left** **Arithmetic Shift Left Index Register X** **Arithmetic Shift Left Index Register Y**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]** **INH** **INH**	**18 78 hh ll** **18 68 xb** **18 68 xb ff** **18 68 xb ee ff** **18 68 xb** **18 68 xb ee ff** **18 48** **18 58**	**ORPWO** **ORPW** **ORPWO** **OfRPWP** **OfIfRPW** **OfIPRPW** **OO** **OO**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	– – – –	Δ Δ Δ Δ

Table A-1. Instruction Set Summary (Sheet 3 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
ASR oprx0_xysp ASR oprx9,xysp ASR oprx16,xysp ASR [D,xysp] ASR [oprx16,xysp] ASRA ASRB	 Arithmetic Shift Right Arithmetic Shift Right Accumulator A Arithmetic Shift Right Accumulator B	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	77 hh ll 67 xb 67 xb ff 67 xb ee ff 67 xb 67 xb ee ff 47 57	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	----	Δ Δ Δ Δ
ASRW oprx0_xysp **ASRW oprx9,xysp** **ASRW oprx16,xysp** **ASRW [D,xysp]** **ASRW [oprx16,xysp]** **ASRX** **ASRY**	 **Arithmetic Shift Right** **Arithmetic Shift Right Index Register X** **Arithmetic Shift Right Index Register Y**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]** **INH** **INH**	**18 77 hh ll** **18 67 xb** **18 67 xb ff** **18 67 xb ee ff** **18 67 xb** **18 67 xb ee ff** **18 47** **18 57**	**ORPWO** **ORPW** **ORPWO** **OfRPWP** **OfIfRPW** **OfIPRPW** **OO** **OO**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ
BCC rel8	Branch if Carry Clear (if C = 0)	REL	24 rr	PPP/P[1]	PPP/P[1]	----	----
BCLR opr8a, msk8 BCLR opr16a, msk8 BCLR oprx0_xysp, msk8 BCLR oprx9,xysp, msk8 BCLR oprx16,xysp, msk8	(M) • ($\overline{mm}$) ⇒ M Clear Bit(s) in Memory	DIR EXT IDX IDX1 IDX2	4D dd mm 1D hh ll mm 0D xb mm 0D xb ff mm 0D xb ee ff mm	rPwO rPwP rPwO rPwP frPwPO	rPwO rPwP rPwO rPwP frPwPO	----	Δ Δ 0 –
BCS rel8	Branch if Carry Set (if C = 1)	REL	25 rr	PPP/P[1]	PPP/P[1]	----	----
BEQ rel8	Branch if Equal (if Z = 1)	REL	27 rr	PPP/P[1]	PPP/P[1]	----	----
BGE rel8	Branch if Greater Than or Equal (if N ⊕ V = 0) (signed)	REL	2C rr	PPP/P[1]	PPP/P[1]	----	----
BGND	Place CPU12 in Background Mode see *CPU12 Reference Manual*	INH	00	VfPPP	VfPPP	----	----
BGT rel8	Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	2E rr	PPP/P[1]	PPP/P[1]	----	----
BHI rel8	Branch if Higher (if C + Z = 0) (unsigned)	REL	22 rr	PPP/P[1]	PPP/P[1]	----	----
BHS rel8	Branch if Higher or Same (if C = 0) (unsigned) same function as BCC	REL	24 rr	PPP/P[1]	PPP/P[1]	----	----
BITA #opr8i BITA opr8a BITA opr16a BITA oprx0_xysp BITA oprx9,xysp BITA oprx16,xysp BITA [D,xysp] BITA [oprx16,xysp]	(A) • (M) Logical AND A with Memory Does not change Accumulator or Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	85 ii 95 dd B5 hh ll A5 xb A5 xb ff A5 xb ee ff A5 xb A5 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ 0 –
BITB #opr8i BITB opr8a BITB opr16a BITB oprx0_xysp BITB oprx9,xysp BITB oprx16,xysp BITB [D,xysp] BITB [oprx16,xysp]	(B) • (M) Logical AND B with Memory Does not change Accumulator or Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C5 ii D5 dd F5 hh ll E5 xb E5 xb ff E5 xb ee ff E5 xb E5 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ 0 –
BITX #opr16i **BITX opr8a** **BITX opr16a** **BITX oprx0_xysp** **BITX oprx9,xysp** **BITX oprx16,xysp** **BITX [D,xysp]** **BITX [oprx16,xysp]**	**(x) • (M:M+1)** **Logical AND X with Memory** **Does not change Index Register or Memory**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 85 jj kk** **18 95 dd** **18 B5 hh ll** **18 A5 xb** **18 A5 xb ff** **18 A5 xb ee ff** **18 A5 xb** **18 A5 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ 0 –
BITY #opr16i **BITY opr8a** **BITY opr16a** **BITY oprx0_xysp** **BITY oprx9,xysp** **BITY oprx16,xysp** **BITY [D,xysp]** **BITY [oprx16,xysp]**	**(Y) • (M:M+1)** **Logical AND Ywith Memory** **Does not change Index Register or Memory**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 C5 jj kk** **18 D5 dd** **18 F5 hh ll** **18 E5 xb** **18 E5 xb ff** **18 E5 xb ee ff** **18 E5 xb** **18 E5 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ 0 –

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

Table A-1. Instruction Set Summary (Sheet 4 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
BLE rel8	Branch if Less Than or Equal (if Z + (N ⊕ V) = 1) (signed)	REL	2F rr	PPP/P[1]	PPP/P[1]	----	----
BLO rel8	Branch if Lower (if C = 1) (unsigned) same function as BCS	REL	25 rr	PPP/P[1]	PPP/P[1]	----	----
BLS rel8	Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	23 rr	PPP/P[1]	PPP/P[1]	----	----
BLT rel8	Branch if Less Than (if N ⊕ V = 1) (signed)	REL	2D rr	PPP/P[1]	PPP/P[1]	----	----
BMI rel8	Branch if Minus (if N = 1)	REL	2B rr	PPP/P[1]	PPP/P[1]	----	----
BNE rel8	Branch if Not Equal (if Z = 0)	REL	26 rr	PPP/P[1]	PPP/P[1]	----	----
BPL rel8	Branch if Plus (if N = 0)	REL	2A rr	PPP/P[1]	PPP/P[1]	----	----
BRA rel8	Branch Always (if 1 = 1)	REL	20 rr	PPP	PPP	----	----
BRCLR opr8a, msk8, rel8 BRCLR opr16a, msk8, rel8 BRCLR oprx0_xysp, msk8, rel8 BRCLR oprx9,xysp, msk8, rel8 BRCLR oprx16,xysp, msk8, rel8	Branch if (M) • (mm) = 0 (if All Selected Bit(s) Clear)	DIR EXT IDX IDX1 IDX2	4F dd mm rr 1F hh ll mm rr 0F xb mm rr 0F xb ff mm rr 0F xb ee ff mm rr	rPPP rfPPP rPPP rfPPP PrfPPP	rPPP rfPPP rPPP rfPPP PrfPPP	----	----
BRN rel8	Branch Never (if 1 = 0)	REL	21 rr	P	P	----	----
BRSET opr8, msk8, rel8 BRSET opr16a, msk8, rel8 BRSET oprx0_xysp, msk8, rel8 BRSET oprx9,xysp, msk8, rel8 BRSET oprx16,xysp, msk8, rel8	Branch if ($\overline{M}$) • (mm) = 0 (if All Selected Bit(s) Set)	DIR EXT IDX IDX1 IDX2	4E dd mm rr 1E hh ll mm rr 0E xb mm rr 0E xb ff mm rr 0E xb ee ff mm rr	rPPP rfPPP rPPP rfPPP PrfPPP	rPPP rfPPP rPPP rfPPP PrfPPP	----	----
BSET opr8, msk8 BSET opr16a, msk8 BSET oprx0_xysp, msk8 BSET oprx9,xysp, msk8 BSET oprx16,xysp, msk8	(M) I (mm) ⇒ M Set Bit(s) in Memory Set CCR flags with respect to the result	DIR EXT IDX IDX1 IDX2	4C dd mm 1C hh ll mm 0C xb mm 0C xb ff mm 0C xb ee ff mm	rPwO rPwP rPwO rPwP frPwPO	rPwO rPwP rPwO rPwP frPwPO	----	ΔΔ0–
BSR rel8	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$ Subroutine address fi PC Branch to Subroutine	REL	07 rr	SPPP	SPPP	----	----
BTAS opr8, msk8 **BTAS opr16a, msk8** **BTAS oprx0_xysp, msk8** **BTAS oprx9,xysp, msk8** **BTAS oprx16,xysp, msk8**	**(M) I (Mask) ⇒ M** **Set Bit(s) in Memory** **Set CCR flags with respect to operand (M) read**	**DIR** **EXT** **IDX** **IDX1** **IDX2**	**18 35 dd mm** **18 36 hh ll mm** **18 37 xb mm** **18 37 xb ff mm** **18 37 xb ee ff mm**	**ORPWO** **ORPWP** **ORPWO** **ORPWP** **OfRPWPO**	**NA** **NA** **NA** **NA** **NA**	----	ΔΔ0–
BVC rel8	Branch if Overflow Bit Clear (if V = 0)	REL	28 rr	PPP/P[1]	PPP/P[1]	----	----
BVS rel8	Branch if Overflow Bit Set (if V = 1)	REL	29 rr	PPP/P[1]	PPP/P[1]	----	----
CALL opr16a, page CALL oprx0_xysp, page CALL oprx9,xysp, page CALL oprx16,xysp, page CALL [D,xysp] CALL [oprx16, xysp]	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$ (SP) – 1 ⇒ SP; (PPG) ⇒ $M_{(SP)}$; pg ⇒ PPAGE register; Program address ⇒ PC Call subroutine in extended memory (Program may be located on another expansion memory page.) Indirect modes get program address and new pg value based on pointer.	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	4A hh ll pg 4B xb pg 4B xb ff pg 4B xb ee ff pg 4B xb 4B xb ee ff	gnSsPPP gnSsPPP gnSsPPP fgnSsPPP fIignSsPPP fIignSsPPP	gnSsPPP gnSsPPP gnSsPPP fgnSsPPP fIignSsPPP fIignSsPPP	----	----
CBA	(A) – (B) Compare 8-Bit Accumulators	INH	18 17	OO	OO	----	Δ Δ Δ Δ
CLC	0 ⇒ C *Translates to* ANDCC #$FE	IMM	10 FE	P	P	----	---0
CLI	0 ⇒ I *Translates to* ANDCC #$EF (enables I-bit interrupts)	IMM	10 EF	P	P	---0	----
CLR opr16a CLR oprx0_xysp CLR oprx9,xysp CLR oprx16,xysp CLR [D,xysp] CLR [oprx16,xysp] CLRA CLRB	0 ⇒ MClear Memory Location 0 ⇒ AClear Accumulator A 0 ⇒ BClear 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	PwO Pw PwO PwP PIfw PIPw O O	----	0100

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

Table A-1. Instruction Set Summary (Sheet 5 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
CLRW oprx16a	**0 ⇒ M:M+1Clear Memory Location**	**EXT**	**18 79 hh ll**	**OPWO**	**NA**	----	0100
CLRW oprx0_xysp		**IDX**	**18 69 xb**	**OPW**	**NA**		
CLRW oprx9,xysp		**IDX1**	**18 69 xb ff**	**OPWO**	**NA**		
CLRW oprx16,xysp		**IDX2**	**18 69 xb ee ff**	**OPWP**	**NA**		
CLRW [D,xysp]		**[D,IDX]**	**18 69 xb**	**OPIfW**	**NA**		
CLRW [oprx16,xysp]		**[IDX2]**	**18 69 xb ee ff**	**OPIPW**	**NA**		
CLRX	**0 ⇒ XClear Index Register X**	**INH**	**18 87**	**OO**	**NA**		
CLRY	**0 ⇒ YClear Index Register Y**	**INH**	**18 C7**	**OO**	**NA**		
CLV	0 ⇒ V *Translates to* ANDCC #$FD	IMM	10 FD	P	P	----	--0-
CMPA #opr8i	(A) – (M)	IMM	81 ii	P	P	----	Δ Δ Δ Δ
CMPA opr8a	Compare Accumulator A with Memory	DIR	91 dd	rPf	rPf		
CMPA opr16a		EXT	B1 hh ll	rPO	rPO		
CMPA oprx0_xysp		IDX	A1 xb	rPf	rPf		
CMPA oprx9,xysp		IDX1	A1 xb ff	rPO	rPO		
CMPA oprx16,xysp		IDX2	A1 xb ee ff	frPP	frPP		
CMPA [D,xysp]		[D,IDX]	A1 xb	fIfrPf	fIfrPf		
CMPA [oprx16,xysp]		[IDX2]	A1 xb ee ff	fIPrPf	fIPrPf		
CMPB #opr8i	(B) – (M)	IMM	C1 ii	P	P	----	Δ Δ Δ Δ
CMPB opr8a	Compare Accumulator B with Memory	DIR	D1 dd	rPf	rPf		
CMPB opr16a		EXT	F1 hh ll	rPO	rPO		
CMPB oprx0_xysp		IDX	E1 xb	rPf	rPf		
CMPB oprx9,xysp		IDX1	E1 xb ff	rPO	rPO		
CMPB oprx16,xysp		IDX2	E1 xb ee ff	frPP	frPP		
CMPB [D,xysp]		[D,IDX]	E1 xb	fIfrPf	fIfrPf		
CMPB [oprx16,xysp]		[IDX2]	E1 xb ee ff	fIPrPf	fIPrPf		
COM opr16a	($\overline{M}$) ⇒ M *equivalent to* $FF – (M) ⇒ M	EXT	71 hh ll	rPwO	rPwO	----	ΔΔ01
COM oprx0_xysp	1's Complement Memory Location	IDX	61 xb	rPw	rPw		
COM oprx9,xysp		IDX1	61 xb ff	rPwO	rPwO		
COM oprx16,xysp		IDX2	61 xb ee ff	frPwP	frPwP		
COM [D,xysp]		[D,IDX]	61 xb	fIfrPw	fIfrPw		
COM [oprx16,xysp]	($\overline{A}$) ⇒ AComplement Accumulator A	[IDX2]	61 xb ee ff	fIPrPw	fIPrPw		
COMA		INH	41	O	O		
COMB	($\overline{B}$) ⇒ BComplement Accumulator B	INH	51	O	O		
COMW oprx16a	**($\overline{M:M+1}$) ⇒ M:M+1 equivalent to**	**EXT**	**18 71 hh ll**	**ORPWO**	**NA**	----	ΔΔ01
COMW oprx0_xysp	**$FF – (M:M+1) ⇒ M:M+1**	**IDX**	**18 61 xb**	**ORPW**	**NA**		
COMW oprx9,xysp		**IDX1**	**18 61 xb ff**	**ORPWO**	**NA**		
COMW oprx16,xysp		**IDX2**	**18 61 xb ee ff**	**OfRPWP**	**NA**		
COMW [D,xysp]		**[D,IDX]**	**18 61 xb**	**OfIfRPW**	**NA**		
COMW [oprx16,xysp]	**($\overline{X}$) ⇒ XComplement Index Register X**	**[IDX2]**	**18 61 xb ee ff**	**OfIPRPW**	**NA**		
COMX		**INH**	**18 41**	**OO**	**NA**		
COMY	**($\overline{Y}$) ⇒ YComplement Index Register Y**	**INH**	**18 51**	**OO**	**NA**		
CPD #opr16i	(A:B) – (M:M+1)	IMM	8C jj kk	PO	PO	----	Δ Δ Δ Δ
CPD opr8a	Compare D to Memory (16-Bit)	DIR	9C dd	RPf	RPf		
CPD opr16a		EXT	BC hh ll	RPO	RPO		
CPD oprx0_xysp		IDX	AC xb	RPf	RPf		
CPD oprx9,xysp		IDX1	AC xb ff	RPO	RPO		
CPD oprx16,xysp		IDX2	AC xb ee ff	fRPP	fRPP		
CPD [D,xysp]		[D,IDX]	AC xb	fIfRPf	fIfRPf		
CPD [oprx16,xysp]		[IDX2]	AC xb ee ff	fIPRPf	fIPRPf		
CPED #opr16i	**(A:B) – (M:M+1) – C**	**IMM**	**18 8C jj kk**	**OPO**	**NA**	----	Δ Δ Δ Δ
CPED opr8a	**Compare D to Memory with Borrow**	**DIR**	**18 9C dd**	**ORPf**	**NA**		
CPED opr16a		**EXT**	**18 BC hh ll**	**ORPO**	**NA**		
CPED oprx0_xysp		**IDX**	**18 AC xb**	**ORPf**	**NA**		
CPED oprx9,xysp		**IDX1**	**18 AC xb ff**	**ORPO**	**NA**		
CPED oprx16,xysp		**IDX2**	**18 AC xb ee ff**	**OfRPP**	**NA**		
CPED [D,xysp]		**[D,IDX]**	**18 AC xb**	**OfIfRPf**	**NA**		
CPED [oprx16,xysp]		**[IDX2]**	**18 AC xb ee ff**	**OfIPRPf**	**NA**		
CPES #opr16i	**(SP) – (M:M+1) – C**	**IMM**	**18 8F jj kk**	**OPO**	**NA**	----	Δ Δ Δ Δ
CPES opr8a	**Compare SP to Memory with Borrow**	**DIR**	**18 9F dd**	**ORPf**	**NA**		
CPES opr16a		**EXT**	**18 BF hh ll**	**ORPO**	**NA**		
CPES oprx0_xysp		**IDX**	**18 AF xb**	**ORPf**	**NA**		
CPES oprx9,xysp		**IDX1**	**18 AF xb ff**	**ORPO**	**NA**		
CPES oprx16,xysp		**IDX2**	**18 AF xb ee ff**	**OfRPP**	**NA**		
CPES [D,xysp]		**[D,IDX]**	**18 AF xb**	**OfIfRPf**	**NA**		
CPES [oprx16,xysp]		**[IDX2]**	**18 AF xb ee ff**	**OfIPRPf**	**NA**		

Table A-1. Instruction Set Summary (Sheet 6 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
CPEX #opr16i **CPEX opr8a** **CPEX opr16a** **CPEX oprx0_xysp** **CPEX oprx9,xysp** **CPEX oprx16,xysp** **CPEX [D,xysp]** **CPEX [oprx16,xysp]**	**(X) – (M:M+1) – C** **Compare X to Memory with Borrow**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 8E jj kk** **18 9E dd** **18 BE hh ll** **18 AE xb** **18 AE xb ff** **18 AE xb ee ff** **18 AE xb** **18 AE xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ
CPEY #opr16i **CPEY opr8a** **CPEY opr16a** **CPEY oprx0_xysp** **CPEY oprx9,xysp** **CPEY oprx16,xysp** **CPEY [D,xysp]** **CPEY [oprx16,xysp]**	**(Y) – (M:M+1) – C** **Compare Y to Memory with Borrow**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 8D jj kk** **18 9D dd** **18 BD hh ll** **18 AD xb** **18 AD xb ff** **18 AD xb ee ff** **18 AD xb** **18 AD xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ
CPS #opr16i CPS opr8a CPS opr16a CPS oprx0_xysp CPS oprx9,xysp CPS oprx16,xysp CPS [D,xysp] CPS [oprx16,xysp]	(SP) – (M:M+1) Compare SP to Memory (16-Bit)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8F jj kk 9F dd BF hh ll AF xb AF xb ff AF xb ee ff AF xb AF xb ee ff	PO RPf RPO RPf RPO fRPP fIfRPf fIPRPf	PO RPf RPO RPf RPO fRPP fIfRPf fIPRPf	----	Δ Δ Δ Δ
CPX #opr16i CPX opr8a CPX opr16a CPX oprx0_xysp CPX oprx9,xysp CPX oprx16,xysp CPX [D,xysp] CPX [oprx16,xysp]	(X) – (M:M+1) Compare X to Memory (16-Bit)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8E jj kk 9E dd BE hh ll AE xb AE xb ff AE xb ee ff AE xb AE xb ee ff	PO RPf RPO RPf RPO fRPP fIfRPf fIPRPf	PO RPf RPO RPf RPO fRPP fIfRPf fIPRPf	----	Δ Δ Δ Δ
CPY #opr16i CPY opr8a CPY opr16a CPY oprx0_xysp CPY oprx9,xysp CPY oprx16,xysp CPY [D,xysp] CPY [oprx16,xysp]	(Y) – (M:M+1) Compare Y to Memory (16-Bit)	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8D jj kk 9D dd BD hh ll AD xb AD xb ff AD xb ee ff AD xb AD xb ee ff	PO RPf RPO RPf RPO fRPP fIfRPf fIPRPf	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	OfO	----	Δ Δ ? Δ
DBEQ abdxys, rel9	(cntr) – 1 ⇒ cntr if (cntr) = 0, then Branch else Continue to next instruction Decrement Counter and Branch if = 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP (branch) PPO (no branch)	----	----
DBNE abdxys, rel9	(cntr) – 1 ⇒ cntr If (cntr) not = 0, then Branch; else Continue to next instruction Decrement Counter and Branch if = 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP (branch) PPO (no branch)	----	----
DEC opr16a DEC oprx0_xysp DEC oprx9,xysp DEC oprx16,xysp DEC [D,xysp] DEC [oprx16,xysp] 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 ll 63 xb 63 xb ff 63 xb ee ff 63 xb 63 xb ee ff 43 53	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	----	Δ Δ Δ –
DECW oprx16a **DECW oprx0_xysp** **DECW oprx9,xysp** **DECW oprx16,xysp** **DECW [D,xysp]** **DECW [oprx16,xysp]** **DECX** **DECY**	**(M:M+1) – $01 ⇒ M:M+1** **Decrement Memory Location** **(X) – $01 ⇒ X Decrement X** **(Y) – $01 ⇒ Y Decrement Y**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]** **INH** **INH**	**18 73 hh ll** **18 63 xb** **18 63 xb ff** **18 63 xb ee ff** **18 63 xb** **18 63 xb ee ff18** **18 43** **18 53**	**ORPWO** **ORPW** **ORPWO** **OfRPWP** **OfIfRPW** **OfIPRPW** **OO** **OO**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ –

Table A-1. Instruction Set Summary (Sheet 7 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
DES	(SP) – $0001 ⇒ SP *Translates to* LEAS –1,SP	IDX	1B 9F	Pf	Pf	----	----
DEX	(X) – $0001 ⇒ X Decrement Index Register X	INH	09	O	O	----	-Δ--
DEY	(Y) – $0001 ⇒ Y Decrement Index Register Y	INH	03	O	O	----	-Δ--
EDIV	(Y:D) ÷ (X) ⇒ Y Remainder fi D 32 by 16 Bit ⇒ 16 Bit Divide (unsigned)	INH	11	fffffffffO	fffffffffO	----	Δ Δ Δ Δ
EDIVS	(Y:D) ÷ (X) ⇒ Y Remainder fi D 32 by 16 Bit ⇒ 16 Bit Divide (signed)	INH	18 14	OfffffffffO	OfffffffffO	----	Δ Δ Δ Δ
EMACS oprx16a[1]	($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 ll	ORRORRWPP	ORROfffRRfWWP	----	Δ Δ Δ Δ
EMAXD oprx0_xysp EMAXD oprx9,xysp EMAXD oprx16,xysp EMAXD [D,xysp] EMAXD [oprx16,xysp]	MAX((D), (M:M+1)) ⇒ D MAX of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1A xb 18 1A xb ff 18 1A xb ee ff 18 1A xb 18 1A xb ee ff	ORPf ORPO OfRPP OfIfRPf OfIPRPf	ORPf ORPO OfRPP OfIfRPf OfIPRPf	----	Δ Δ Δ Δ
EMAXM oprx0_xysp EMAXM oprx9,xysp EMAXM oprx16,xysp EMAXM [D,xysp] EMAXM [oprx16,xysp]	MAX((D), (M:M+1)) ⇒ M:M+1 MAX of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1E xb 18 1E xb ff 18 1E xb ee ff 18 1E xb 18 1E xb ee ff	ORPW ORPWO OfRPWP OfIfRPW OfIPRPW	ORPW ORPWO OfRPWP OfIfRPW OfIPRPW	----	Δ Δ Δ Δ
EMIND oprx0_xysp EMIND oprx9,xysp EMIND oprx16,xysp EMIND [D,xysp] EMIND [oprx16,xysp]	MIN((D), (M:M+1)) ⇒ D MIN of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1B xb 18 1B xb ff 18 1B xb ee ff 18 1B xb 18 1B xb ee ff	ORPf ORPO OfRPP OfIfRPf OfIPRPf	ORPf ORPO OfRPP OfIfRPf OfIPRPf	----	Δ Δ Δ Δ
EMINM oprx0_xysp EMINM oprx9,xysp EMINM oprx16,xysp EMINM [D,xysp] EMINM [oprx16,xysp]	MIN((D), (M:M+1)) ⇒ M:M+1 MIN of 2 Unsigned 16-Bit Values N, Z, V and C status bits reflect result of internal compare ((D) – (M:M+1))	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1F xb 18 1F xb ff 18 1F xb ee ff 18 1F xb 18 1F xb ee ff	ORPW ORPWO OfRPWP OfIfRPW OfIPRPW	ORPW ORPWO OfRPWP OfIfRPW OfIPRPW	----	Δ Δ Δ Δ
EMUL	(D) × (Y) ⇒ Y:D 16 by 16 Bit Multiply (unsigned)	INH	13	O	ff0	----	ΔΔ–Δ
EMULS	(D) × (Y) ⇒ Y:D 16 by 16 Bit Multiply (signed)	INH	18 13	OfO (if followed by Page 2 instruction) OffO	OfO OffO	----	ΔΔ–D
EORA #opr8i EORA opr8a EORA opr16a EORA oprx0_xysp EORA oprx9,xysp EORA oprx16,xysp EORA [D,xysp] EORA [oprx16,xysp]	(A) ⊕ (M) ⇒ A Exclusive-OR A with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	88 ii 98 dd B8 hh ll A8 xb A8 xb ff A8 xb ee ff A8 xb A8 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	----	ΔΔ0–
EORB #opr8i EORB opr8a EORB opr16a EORB oprx0_xysp EORB oprx9,xysp EORB oprx16,xysp EORB [D,xysp] EORB [oprx16,xysp]	(B) ⊕ (M) ⇒ 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 rPf rPO rPf rPO frPP fIfrPf fIPrPf	----	ΔΔ0–
EORX #opr16i **EORX opr8a** **EORX opr16a** **EORX oprx0_xysp** **EORX oprx9,xysp** **EORX oprx16,xysp** **EORX [D,xysp]** **EORX [oprx16,xysp]**	**(X) ⊕ (M:M+1) ⇒ X** **Exclusive-OR X with Memory**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 88 jj kk** **18 98 dd** **18 B8 hh ll** **18 A8 xb** **18 A8 xb ff** **18 A8 xb ee ff** **18 A8 xb** **18 A8 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	ΔΔ0–

Note:1. *oprx16a* is an extended address specifiation. Both X and Y point to source operands.

Table A-1. Instruction Set Summary (Sheet 8 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
EORY #opr16i **EORY opr8a** **EORY opr16a** **EORY oprx0_xysp** **EORY oprx9,xysp** **EORY oprx16,xysp** **EORY [D,xysp]** **EORY [oprx16,xysp]**	**(Y) ⊕ (M:M+1) ⇒ Y** **Exclusive-OR Y with Memory**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 C8 jj kk** **18 D8 dd** **18 F8 hh ll** **18 E8 xb** **18 E8 xb ff** **18 E8 xb ee ff** **18 E8 xb** **18 E8 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	---- 	ΔΔ0–
ETBL oprx0_xysp	(M:M+1) + [(B) × ((M+2:M+3) – (M:M+1))] ⇒ D 16-Bit Table Lookup and Interpolate Initialize B, and index before ETBL. <ea> points at first table entry (M:M+1) and B is fractional part of lookup value (no indirect addr. modes or extensions allowed)	IDX	18 3F xb	ORRfffffffP	ORRfffffffP	----	ΔΔ–Δ
EXG abcdxys,abcdxys	(r1) ⇔ (r2) (if r1 and r2 same size) *or* $00:(r1) ⇒ r2 (if r1=8-bit; r2=16-bit) *or* ($r1_{low}$) ⇔ (r2) (if r1=16-bit; r2=8-bit) r1 and r2 may be A, B, CCR, D, X, Y, or SP	INH	B7 eb	P	P	----	----
FDIV	(D) ÷ (X) ⇒ X; Remainder fi D 16 by 16 Bit Fractional Divide	INH	18 11	OfffffffffffO	OfffffffffffO	----	–ΔΔΔ
GLDAA oprx0_xysp **GLDAA oprx9,xysp** **GLDAA oprx16,xysp** **GLDAA [D,xysp]** **GLDAA [oprx16,xysp]**	**G(M) ⇒ A** **Load Accumulator A from Global Memory**	**IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 A6 xb** **18 A6 xb ff** **18 A6 xb ee ff** **18 A6 xb** **18 A6 xb ee ff**	**OrPf** **OrPO** **OfrPP** **OfIfrPf** **OfIPrPf**	**NA** **NA** **NA** **NA** **NA**	---- 	ΔΔ0–
GLDAB oprx0_xysp **GLDAB oprx9,xysp** **GLDAB oprx16,xysp** **GLDAB [D,xysp]** **GLDAB [oprx16,xysp]**	**G(M) ⇒ B** **Load Accumulator B from Global Memory**	**IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 E6 xb** **18 E6 xb ff** **18 E6 xb ee ff** **18 E6 xb** **18 E6 xb ee ff**	**OrPf** **OrPO** **OfrPP** **OfIfrPf** **OfIPrPf**	**NA** **NA** **NA** **NA** **NA**	---- 	ΔΔ0–
GLDD oprx0_xysp **GLDD oprx9,xysp** **GLDD oprx16,xysp** **GLDD [D,xysp]** **GLDD [oprx16,xysp]**	**G(M:M+1) ⇒ A:B** **Load Double Accumulator D (A:B) from Global Memory**	**IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 EC xb** **18 EC xb ff** **18 EC xb ee ff** **18 EC xb** **18 EC xb ee ff**	**ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA**	---- 	ΔΔ0–
GLDS oprx0_xysp **GLDS oprx9,xysp** **GLDS oprx16,xysp** **GLDS [D,xysp]** **GLDS [oprx16,xysp]**	**G(M:M+1) ⇒ SP** **Load Stack Pointer from Global Memory**	**IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 EF xb** **18 EF xb ff** **18 EF xb ee ff** **18 EF xb** **18 EF xb ee ff**	**ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA**	---- 	ΔΔ0–
GLDX oprx0_xysp **GLDX oprx9,xysp** **GLDX oprx16,xysp** **GLDX [D,xysp]** **GLDX [oprx16,xysp]**	**G(M:M+1) ⇒ X** **Load Index Register X from Global Memory**	**IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 EE xb** **18 EE xb ff** **18 EE xb ee ff** **18 EE xb** **18 EE xb ee ff**	**ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA**	---- 	ΔΔ0–
GLDY oprx0_xysp **GLDY oprx9,xysp** **GLDY oprx16,xysp** **GLDY [D,xysp]** **GLDY [oprx16,xysp]**	**G(M:M+1) ⇒ Y** **Load Index Register Y from Global Memory**	**IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 ED xb** **18 ED xb ff** **18 ED xb ee ff** **18 ED xb** **18 ED xb ee ff**	**ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA**	---- 	ΔΔ0–

Table A-1. Instruction Set Summary (Sheet 9 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
GSTAA oprx0_xysp	**(A) ⇒ G(M)**	**DIR**	**18 5A dd**	**OPw**	**NA**	– – – –	Δ Δ 0 –
GSTAA oprx9,xysp	**Store Accumulator A to Global Memory**	**EXT**	**18 7A hh ll**	**OPwO**	**NA**		
GSTAA oprx16,xysp		**IDX**	**18 6A xb**	**OPw**	**NA**		
GSTAA [D,xysp]		**IDX1**	**18 6A xb ff**	**OPwO**	**NA**		
GSTAA [oprx16,xysp]		**IDX2**	**18 6A xb ee ff**	**OPwP**	**NA**		
		[D,IDX]	**18 6A xb**	**OPIfw**	**NA**		
		[IDX2]	**18 6A xb ee ff**	**OPIPw**	**NA**		
GSTAB oprx0_xysp	**(B) ⇒ G(M)**	**DIR**	**18 5B dd**	**OPw**	**NA**	– – – –	Δ Δ 0 –
GSTAB oprx9,xysp	**Store Accumulator B to Global Memory**	**EXT**	**18 7B hh ll**	**OPwO**	**NA**		
GSTAB oprx16,xysp		**IDX**	**18 6B xb**	**OPw**	**NA**		
GSTAB [D,xysp]		**IDX1**	**18 6B xb ff**	**OPwO**	**NA**		
GSTAB [oprx16,xysp]		**IDX2**	**18 6B xb ee ff**	**OPwP**	**NA**		
		[D,IDX]	**18 6B xb**	**OPIfw**	**NA**		
		[IDX2]	**18 6B xb ee ff**	**OPIPw**	**NA**		
GSTD oprx0_xysp	**(A) ⇒ G(M), (B) ⇒ G(M+1)**	**DIR**	**18 5C dd**	**OPW**	**NA**	– – – –	Δ Δ 0 –
GSTD oprx9,xysp	**Store Double Accumulator to Global Memory**	**EXT**	**18 7C hh ll**	**OPWO**	**NA**		
GSTD oprx16,xysp		**IDX**	**18 6C xb**	**OPW**	**NA**		
GSTD [D,xysp]		**IDX1**	**18 6C xb ff**	**OPWO**	**NA**		
GSTD [oprx16,xysp]		**IDX2**	**18 6C xb ee ff**	**OPWP**	**NA**		
		[D,IDX]	**18 6C xb**	**OPIfW**	**NA**		
		[IDX2]	**18 6C xb ee ff**	**OPIPW**	**NA**		
GSTS oprx0_xysp	**(SP) ⇒ G(M:M+1)**	**DIR**	**18 5F dd**	**OPW**	**NA**	– – – –	Δ Δ 0 –
GSTS oprx9,xysp	**Store Stack Pointer to Global Memory**	**EXT**	**18 7F hh ll**	**OPWO**	**NA**		
GSTS oprx16,xysp		**IDX**	**18 6F xb**	**OPW**	**NA**		
GSTS [D,xysp]		**IDX1**	**18 6F xb ff**	**OPWO**	**NA**		
GSTS [oprx16,xysp]		**IDX2**	**18 6F xb ee ff**	**OPWP**	**NA**		
		[D,IDX]	**18 6F xb**	**OPIfW**	**NA**		
		[IDX2]	**18 6F xb ee ff**	**OPIPW**	**NA**		
GSTX oprx0_xysp	**(X) ⇒ G(M:M+1)**	**DIR**	**18 5E dd**	**OPW**	**NA**	– – – –	Δ Δ 0 –
GSTX oprx9,xysp	**Store Index Register X to Global Memory**	**EXT**	**18 7E hh ll**	**OPWO**	**NA**		
GSTX oprx16,xysp		**IDX**	**18 6E xb**	**OPW**	**NA**		
GSTX [D,xysp]		**IDX1**	**18 6E xb ff**	**OPWO**	**NA**		
GSTX [oprx16,xysp]		**IDX2**	**18 6E xb ee ff**	**OPWP**	**NA**		
		[D,IDX]	**18 6E xb**	**OPIfW**	**NA**		
		[IDX2]	**18 6E xb ee ff**	**OPIPW**	**NA**		
GSTY oprx0_xysp	**(Y) ⇒ G(M:M+1)**	**DIR**	**18 5D dd**	**OPW**	**NA**	– – – –	Δ Δ 0 –
GSTY oprx9,xysp	**Store Index Register Yto Global Memory**	**EXT**	**18 7D hh ll**	**OPWO**	**NA**		
GSTY oprx16,xysp		**IDX**	**18 6D xb**	**OPW**	**NA**		
GSTY [D,xysp]		**IDX1**	**18 6D xb ff**	**OPWO**	**NA**		
GSTY [oprx16,xysp]		**IDX2**	**18 6D xb ee ff**	**OPWP**	**NA**		
		[D,IDX]	**18 6D xb**	**OPIfW**	**NA**		
		[IDX2]	**18 6D xb ee ff**	**OPIPW**	**NA**		
IBEQ abdxys, rel9	(cntr) + 1 ⇒ cntr If (cntr) = 0, then Branch else Continue to next instruction Increment Counter and Branch if = 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP (branch) PPO (no branch)	– – – –	– – – –
IBNE abdxys, rel9	(cntr) + 1 ⇒ cntr if (cntr) not = 0, then Branch; else Continue to next instruction Increment Counter and Branch if ≠ 0 (cntr = A, B, D, X, Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP (branch) PPO (no branch)	– – – –	– – – –
IDIV	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Integer Divide (unsigned)	INH	18 10	OfffffffffO	OfffffffffO	– – – –	– Δ 0 Δ
IDIVS	(D) ÷ (X) ⇒ X; Remainder ⇒ D 16 by 16 Bit Integer Divide (signed)	INH	18 15	OfffffffffO	OfffffffffO	– – – –	Δ Δ Δ Δ
INC oprx16a	(M) + $01 ⇒ M	EXT	72 hh ll	rPwO	rPwO	– – – –	Δ Δ Δ –
INC oprx0_xysp	Increment Memory Byte	IDX	62 xb	rPw	rPw		
INC oprx9,xysp		IDX1	62 xb ff	rPwO	rPwO		
INC oprx16,xysp		IDX2	62 xb ee ff	frPwP	frPwP		
INC [D,xysp]		[D,IDX]	62 xb	fIfrPw	fIfrPw		
INC [oprx16,xysp]		[IDX2]	62 xb ee ff	fIPrPw	fIPrPw		
INCA	(A) + $01 ⇒ AIncrement Acc. A	INH	42	O	O		
INCB	(B) + $01 ⇒ BIncrement Acc. B	INH	52	O	O		

Table A-1. Instruction Set Summary (Sheet 10 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
INCW oprx16a **INCW oprx0_xysp** **INCW oprx9,xysp** **INCW oprx16,xysp** **INCW [D,xysp]** **INCW [oprx16,xysp]** **INCX** **INCY**	**(M:M+1) + $01 ⇒ M:M+1** **Increment Memory** **(X) + $01 ⇒ XIncrement Index Register X** **(Y) + $01 ⇒ YIncrement Index Register Y**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]** **INH** **INH**	**18 72 hh ll** **18 62 xb** **18 62 xb ff** **18 62 xb ee ff** **18 62 xb** **18 62 xb ee ff** **18 42** **18 52**	**ORPWO** **ORPW** **ORPWO** **OfRPWP** **OfIfRPW** **OfIPRPW** **OO** **OO**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	ΔΔΔ–
INS	(SP) + $0001 ⇒ SP *Translates to* LEAS 1,SP	IDX	1B 81	Pf	Pf	----	----
INX	(X) + $0001 ⇒ X Increment Index Register X	INH	08	O	O	----	–Δ––
INY	(Y) + $0001 ⇒ Y Increment Index Register Y	INH	02	O	O	----	–Δ––
JMP oprx16a JMP oprx0_xysp JMP oprx9,xysp JMP oprx16,xysp JMP [D,xysp] JMP [oprx16,xysp]	Routine address ⇒ PC Jump	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	06 hh ll 05 xb 05 xb ff 05 xb ee ff 05 xb 05 xb ee ff	PPP PPP PPP fPPP fIfPPP fIfPPP	PPP PPP PPP fPPP fIfPPP fIfPPP	----	----
JSR oprx8a JSR oprx16a JSR oprx0_xysp JSR oprx9,xysp JSR oprx16,xysp JSR [D,xysp] JSR [oprx16,xysp]	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$; Subroutine address ⇒ PC Jump to Subroutine	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	17 dd 16 hh ll 15 xb 15 xb ff 15 xb ee ff 15 xb 15 xb ee ff	SPPP SPPP PPPS PPPS fPPPS fIfPPPS fIfPPPS	SPPP SPPP PPPS PPPS fPPPS fIfPPPS fIfPPPS	----	----
LBCC rel16	Long Branch if Carry Clear (if C = 0)	REL	18 24 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBCS rel16	Long Branch if Carry Set (if C = 1)	REL	18 25 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBEQ rel16	Long Branch if Equal (if Z = 1)	REL	18 27 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBGE rel16	Long Branch Greater Than or Equal (if N ⊕ V = 0) (signed)	REL	18 2C qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBGT rel16	Long Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	18 2E qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBHI rel16	Long Branch if Higher (if C + Z = 0) (unsigned)	REL	18 22 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBHS rel16	Long Branch if Higher or Same (if C = 0) (unsigned) same function as LBCC	REL	18 24 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBLE rel16	Long Branch if Less Than or Equal (if Z + (N ⊕ V) = 1) (signed)	REL	18 2F qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBLO rel16	Long Branch if Lower (if C = 1) (unsigned) *same function as LBCS*	REL	18 25 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBLS rel16	Long Branch if Lower or Same (if C + Z = 1) (unsigned)	REL	18 23 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBLT rel16	Long Branch if Less Than (if N ⊕ V = 1) (signed)	REL	18 2D qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBMI rel16	Long Branch if Minus (if N = 1)	REL	18 2B qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBNE rel16	Long Branch if Not Equal (if Z = 0)	REL	18 26 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBPL rel16	Long Branch if Plus (if N = 0)	REL	18 2A qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBRA rel16	Long Branch Always (if 1 = 1)	REL	18 20 qq rr	OPPP	OPPP	----	----
LBRN rel16	Long Branch Never (if 1 = 0)	REL	18 21 qq rr	OPO	OPO	----	----
LBVC rel16	Long Branch if Overflow Bit Clear (if V = 0)	REL	18 28 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----
LBVS rel16	Long Branch if Overflow Bit Set (if V = 1)	REL	18 29 qq rr	OPPP/OPO[1]	OPPP/OPO[1]	----	----

LDAA #opr8i
LDAA opr8a
LDAA opr16a
LDAA oprx0_xysp
LDAA oprx9,xysp
LDAA oprx16,xysp
LDAA [D,xysp]
LDAA [oprx16,xysp]
	(M) ⇒ A
	Load Accumulator A
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			86 ii
			96 dd
			B6 hh ll
			A6 xb
			A6 xb ff
			A6 xb ee ff
			A6 xb
			A6 xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rPf
					rPO
					rPf
					rPO
					frPP
					fIfrPf
					fIPrPf
						----	ΔΔ0–

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

Table A-1. Instruction Set Summary (Sheet 11 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C

LDAB #opr8i
LDAB opr8a
LDAB opr16a
LDAB oprx0_xysp
LDAB oprx9,xysp
LDAB oprx16,xysp
LDAB [D,xysp]
LDAB [oprx16,xysp]
	(M) ⇒ B
	Load Accumulator B
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			C6 ii
			D6 dd
			F6 hh ll
			E6 xb
			E6 xb ff
			E6 xb ee ff
			E6 xb
			E6 xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rPf
					rPO
					rPf
					rPO
					frPP
					fIfrPf
					fIPrPf
						----	ΔΔ0–

LDD #opr16i
LDD opr8a
LDD opr16a
LDD oprx0_xysp
LDD oprx9,xysp
LDD oprx16,xysp
LDD [D,xysp]
LDD [oprx16,xysp]
	(M:M+1) ⇒ A:B
	Load Double Accumulator D (A:B)
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			CC jj kk
			DC dd
			FC hh ll
			EC xb
			EC xb ff
			EC xb ee ff
			EC xb
			EC xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					PO
					RPf
					RPO
					RPf
					RPO
					fRPP
					fIfRPf
					fIPRPf
						----	ΔΔ0–

LDS #opr16i
LDS opr8a
LDS opr16a
LDS oprx0_xysp
LDS oprx9,xysp
LDS oprx16,xysp
LDS [D,xysp]
LDS [oprx16,xysp]
	(M:M+1) ⇒ SP
	Load Stack Pointer
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			CF jj kk
			DF dd
			FF hh ll
			EF xb
			EF xb ff
			EF xb ee ff
			EF xb
			EF xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					PO
					RPf
					RPO
					RPf
					RPO
					fRPP
					fIfRPf
					fIPRPf
						----	ΔΔ0–

LDX #opr16i
LDX opr8a
LDX opr16a
LDX oprx0_xysp
LDX oprx9,xysp
LDX oprx16,xysp
LDX [D,xysp]
LDX [oprx16,xysp]
	(M:M+1) ⇒ X
	Load Index Register X
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			CE jj kk
			DE dd
			FE hh ll
			EE xb
			EE xb ff
			EE xb ee ff
			EE xb
			EE xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					PO
					RPf
					RPO
					RPf
					RPO
					fRPP
					fIfRPf
					fIPRPf
						----	ΔΔ0–

LDY #opr16i
LDY opr8a
LDY opr16a
LDY oprx0_xysp
LDY oprx9,xysp
LDY oprx16,xysp
LDY [D,xysp]
LDY [oprx16,xysp]
	(M:M+1) ⇒ Y
	Load Index Register Y
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			CD jj kk
			DD dd
			FD hh ll
			ED xb
			ED xb ff
			ED xb ee ff
			ED xb
			ED xb ee ff
				PO
				RPf
				RPO
				RPf
				RPO
				fRPP
				fIfRPf
				fIPRPf
					PO
					RPf
					RPO
					RPf
					RPO
					fRPP
					fIfRPf
					fIPRPf
						----	ΔΔ0–

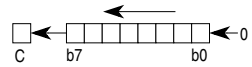
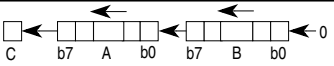
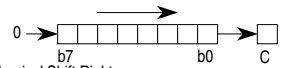
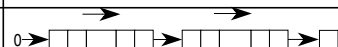
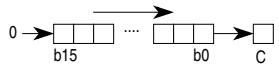
LEAS oprx0_xysp LEAS oprx9,xysp LEAS oprx16,xysp	Effective Address ⇒ SP Load Effective Address into SP	IDX IDX1 IDX2	1B xb 1B xb ff 1B xb ee ff	Pf PO PP	Pf PO PP	----	----
LEAX oprx0_xysp LEAX oprx9,xysp LEAX oprx16,xysp	Effective Address ⇒ X Load Effective Address into X	IDX IDX1 IDX2	1A xb 1A xb ff 1A xb ee ff	Pf PO PP	Pf 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	Pf PO PP	----	----
LSL opr16a LSL oprx0_xysp LSL oprx9,xysp LSL oprx16,xysp LSL [D,xysp] LSL [oprx16,xysp] LSLA LSLB	 Logical Shift Left same function as ASL Logical Shift Accumulator A to Left Logical Shift Accumulator B to Left	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	78 hh ll 68 xb 68 xb ff 68 xb ee ff 68 xb 68 xb ee ff 48 58	rPwO rPw rPwO frPPw fIfrPw fIPrPw O O	rPwO rPw rPwO frPPw fIfrPw fIPrPw O O	----	Δ Δ Δ Δ
LSLD	 Logical Shift Left D Accumulator same function as ASLD	INH	59	O	O	----	Δ Δ Δ Δ
LSR opr16a LSR oprx0_xysp LSR oprx9,xysp LSR oprx16,xysp LSR [D,xysp] LSR [oprx16,xysp] LSRA LSRB	 Logical Shift Right Logical Shift Accumulator A to Right Logical Shift Accumulator B to Right	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	74 hh ll 64 xb 64 xb ff 64 xb ee ff 64 xb 64 xb ee ff 44 54	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	----	0 Δ Δ Δ

Table A-1. Instruction Set Summary (Sheet 12 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
LSRD	 Logical Shift Right D Accumulator	INH	49	O	O	----	0 Δ Δ Δ
LSRW oprx0_xysp **LSRW oprx9,xysp** **LSRW oprx16,xysp** **LSRW [D,xysp]** **LSRW [oprx16,xysp]** **LSRX** **LSRY**	 **Logical Shift Index Register X to Right** **Logical Shift Index Register Y to Right**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]** **INH** **INH**	**18 74 hh ll** **18 64 xb** **18 64 xb ff** **18 64 xb ee ff** **18 64 xb** **18 64 xb ee ff** **18 44** **18 54**	**ORPWO** **ORPW** **ORPWO** **OfRPWP** **OfIfRPW** **OfIPRPW** **OO** **OO**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	0 Δ Δ Δ
MAXA oprx0_xysp MAXA oprx9,xysp MAXA oprx16,xysp MAXA [D,xysp] MAXA [oprx16,xysp]	MAX((A), (M)) ⇒ A MAX of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) – (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 18 xb 18 18 xb ff 18 18 xb ee ff 18 18 xb 18 18 xb ee ff	OrPf OrPO OfrPP OfIfrPf OfIPrPf	OrPf OrPO OfrPP OfIfrPf OfIPrPf	----	Δ Δ Δ Δ
MAXM oprx0_xysp MAXM oprx9,xysp MAXM oprx16,xysp MAXM [D,xysp] MAXM [oprx16,xysp]	MAX((A), (M)) ⇒ M MAX of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) – (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1C xb 18 1C xb ff 18 1C xb ee ff 18 1C xb 18 1C xb ee ff	OrPw OrPwO OfrPwP OfIfrPw OfIPrPw	OrPw OrPwO OfrPwP OfIfrPw OfIPrPw	----	Δ Δ Δ Δ
MEM	m (grade) fi $M_{(Y)}$; (X) + 4 ⇒ X; (Y) + 1 ⇒ Y; A unchanged if (A) < P1 or (A) > P2 then m = 0, else m = MIN[((A) – P1) × S1, (P2 – (A)) × S2, $FF] where: A = current crisp input value; X points at 4-byte data structure that describes a trapezoidal membership function (P1, P2, S1, S2); Y points at fuzzy input (RAM location). See *CPU12 Reference Manual* for special cases.	Special	01	RRfOw	RRfOw	--?-	????
MINA oprx0_xysp MINA oprx9,xysp MINA oprx16,xysp MINA [D,xysp] MINA [oprx16,xysp]	MIN((A), (M)) ⇒ A MIN of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) – (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 19 xb 18 19 xb ff 18 19 xb ee ff 18 19 xb 18 19 xb ee ff	OrPf OrPO OfrPP OfIfrPf OfIPrPf	OrPf OrPO OfrPP OfIfrPf OfIPrPf	----	Δ Δ Δ Δ
MINM oprx0_xysp MINM oprx9,xysp MINM oprx16,xysp MINM [D,xysp] MINM [oprx16,xysp]	MIN((A), (M)) ⇒ M MIN of 2 Unsigned 8-Bit Values N, Z, V and C status bits reflect result of internal compare ((A) – (M)).	IDX IDX1 IDX2 [D,IDX] [IDX2]	18 1D xb 18 1D xb ff 18 1D xb ee ff 18 1D xb 18 1D xb ee ff	OrPw OrPwO OfrPwP OfIfrPw OfIPrPw	OrPw OrPwO OfrPwP OfIfrPw OfIPrPw	----	Δ Δ Δ Δ
MOVB #opr8i, opr16a[1] **MOVB #opr8i, oprx0_xysp**[1] **MOVB #opr8i, oprx9_xysp**[1] **MOVB #opr8i, oprx16_xysp**[1] **MOVB #opr8i, [D_xysp]**[1] **MOVB #opr8i, [oprx16_xysp]**[1]	**# ⇒ M** **Immediate to Memory Byte-Move (8-Bit)**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 0B ii hh ll** **18 08 xb**[2] **ii** **18 08 xb**[2] **ff ii** **18 08 xb**[2] **ee ff ii** **18 08 xb**[2] **ii** **18 08 xb**[2] **ee ff ii**	PwP PwO PwP PPwO PIOw PIOwP	**NA** **NA** **NA** **NA** **NA** **NA**	----	----
MOVB oprx16a, oprx16a[1] **MOVB oprx16a, oprx0_xysp**[1] **MOVB oprx16a, oprx9_xysp**[1] **MOVB oprx16a, oprx16_xysp**[1] **MOVB oprx16a, [D_xysp]**[1] **MOVB oprx16a, [oprx16_xysp]**[1]	**(M_1) ⇒ M_2** **Memory to Memory Byte-Move (8-Bit)** **EXT Source fi Addr. Mode Destination**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 0C hh ll hh ll** **18 09 xb**[2] **hh ll** **18 09 xb**[2] **ff hh ll** **18 09 xb**[2] **ee ff hh ll** **18 09 xb**[2] **hh ll** **18 09 xb**[2] **ee ff hh ll**	PrPwO PrPw PrPwO PPrPw PrIPw PPrIPw	**NA** **NA** **NA** **NA** **NA** **NA**	----	----
MOVB oprx0_xysp, opr16a[1] **MOVB oprx0_xysp, oprx0_xysp**[1] **MOVB oprx0_xysp, oprx9_xysp**[1] **MOVB oprx0_xysp, oprx16_xysp**[1] **MOVB oprx0_xysp, [D_xysp]**[1] **MOVB oprx0_xysp, [oprx16_xysp]**[1]	**(M_1) ⇒ M_2** **Memory to Memory Byte-Move (8-Bit)** **IDX Source fi Addr. Mode Destination**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 0D xb hh ll** **18 0A xb xb** **18 0A xbxb ff** **18 0A xb xb ee ff** **18 0A xb xb** **18 0A xb xb ee ff**	rPPw rPOw rPPw rPOPw rPIOw rPPIOw	**NA** **NA** **NA** **NA** **NA** **NA**	----	----
MOVB oprx9_xysp, opr16a[1] **MOVB oprx9_xysp, oprx0_xysp**[1] **MOVB oprx9_xysp, oprx9_xysp**[1] **MOVB oprx9_xysp, oprx16_xysp**[1] **MOVB oprx9_xysp, [D_xysp]**[1] **MOVB oprx9_xysp, [oprx16_xysp]**[1]	**(M_1) ⇒ M_2** **Memory to Memory Byte-Move (8-Bit),** **IDX1 Source fi Addr. Mode Destination**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 0D xb ff hh ll** **18 0A xb ff xb** **18 0A xb ff xb ff** **18 0A xb ff xb ee ff** **18 0A xb ff xb** **18 0A xb ff xb ee ff**	PrOPw PrOOw PrOPw PrOOPw PrOIOw PrOPIOw	**NA** **NA** **NA** **NA** **NA** **NA**	----	----

Notes: 1. The first operand in the source code statement specifies the source for the move.
2. The IDX destination code is listed before the source for backwards compatibility.

Table A-1. Instruction Set Summary (Sheet 13 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C

MOVB oprx16_xysp, opr16a[1]
MOVB oprx16_xysp, oprx0_xysp[1]
MOVB oprx16_xysp, oprx9_xysp[1]
MOVB oprx16_xysp, oprx16_xysp[1]
MOVB oprx16_xysp, [D_xysp][1]
MOVB oprx16_xysp, [oprx16_xysp][1]
	$(M_1) \Rightarrow M_2$
	Memory to Memory Byte-Move (8-Bit),
	IDX2 Source fi Addr. Mode Destination
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			18 0D xb ee ff hh ll
			18 0A xb ee ff xb
			18 0A xb ee ff xb ff
			18 0A xb ee ff xb ee ff
			18 0A xb ee ff xb
			18 0A xb ee ff xb ee ff
				PrPPw
				PrPOw
				
				PrPPw
				
				PrPOPw
				
				PrPIOw
				
				PrPPIO
					NA
					NA
					NA
					NA
					NA
					NA
						----	----

MOVB [D_xysp], opr16a[1]
MOVB [D_xysp], oprx0_xysp[1]
MOVB [D_xysp], oprx9_xysp[1]
MOVB [D_xysp], oprx16_xysp[1]
MOVB [D_xysp], [D_xysp][1]
MOVB [D_xysp], [oprx16_xysp][1]
	$(M_1) \Rightarrow M_2$
	Memory to Memory Byte-Move (8-Bit),
	[D,IDX] Source fi Addr. Mode Destination
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			18 0D xb hh ll
			18 0A xb xb
			18 0A xb xb ff
			18 0A xb xb ee ff
			18 0A xb xb
			18 0A xb xb ee ff
				IPrfPw
				IPrfOw
				IPrfPw
				IPrfOPw
				IPrfIOw
				IPrfPIOw
					NA
					NA
					NA
					NA
					NA
					NA
						----	----

MOVB [oprx16_xysp], opr16a[1]
MOVB [oprx16_xysp], oprx0_xysp[1]
MOVB [oprx16_xysp], oprx9_xysp[1]
MOVB [oprx16_xysp], oprx16_xysp[1]
MOVB [oprx16_xysp], [D_xysp][1]
MOVB [oprx16_xysp], [oprx16_xysp][1]
	$(M_1) \Rightarrow M_2$
	Memory to Memory Byte-Move (8-Bit),
	[IDX2] Source fi Addr. Mode Destination
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			18 0D xb ee ff hh ll
			18 0A xb ee ff xb
			18 0A xb ee ff xb ff
			18 0A xb ee ff xb ee ff
			18 0A xb ee ff xb
			18 0A xb ee ff xb ee ff
				PIPrfPw
				PIPrfOw
				
				PIPrfPw
				
				PIPrfOPw
				
				PIPrfIOw
				
				PIPrfPIOw
					NA
					NA
					NA
					NA
					NA
					NA
						----	----

MOVW #opr16i, opr16a[1]
MOVW #opr16i, oprx0_xysp[1]
MOVW #opr16i, oprx9_xysp[1]
MOVW #opr16i, oprx16_xysp[1]
MOVW #opr16i, [D_xysp][1]
MOVW #opr16i, [oprx16_xysp][1]
	$\# \Rightarrow M{:}M+1_2$
	Immediate to Memory Word-Move (16-Bit)
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			18 03 jj kk hh ll
			18 00 xb[2] jj kk
			18 00 xb[2] ff jj kk
			18 00 xb[2] ee ff jj kk
			18 00 xb[2] jj kk
			18 00 xb[2] ee ff jj kk
				PWPO
				PWP
				PPWO
				PPWP
				PIPW
				PIPWP
					NA
					NA
					NA
					NA
					NA
					NA
						----	----

MOVW opr16a, opr16a[1]
MOVW opr16a, oprx0_xysp[1]
MOVW opr16a, oprx9_xysp[1]
MOVW opr16a, oprx16_xysp[1]
MOVW opr16a, [D_xysp][1]
MOVW opr16a, [oprx16_xysp][1]
	$(M{:}M+1_1) \Rightarrow M{:}M+1_2$
	Memory to Memory Word-Move (16-Bit),
	EXT Source fi Addr. Mode Destination
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			18 04 hh ll hh ll
			18 01 xb[2] hh ll
			18 01 xb[2] ff hh ll
			18 01 xb[2] ee ff hh ll
			18 01 xb[2] hh ll
			18 01 xb[2] ee ff hh ll
				PRPWO
				PRPW
				PRPWO
				PPRPW
				PRIPW
				PPRIPW
					NA
					NA
					NA
					NA
					NA
					NA
						----	----

MOVW oprx0_xysp, opr16a[1]
MOVW oprx0_xysp, oprx0_xysp[1]
MOVW oprx0_xysp, oprx9_xysp[1]
MOVW oprx0_xysp, oprx16_xysp[1]
MOVW oprx0_xysp, [D_xysp][1]
MOVW oprx0_xysp, [oprx16_xysp][1]
	$(M{:}M+1_1) \Rightarrow M{:}M+1_2$
	Memory to Memory Word-Move (16-Bit),
	IDX Source fi Addr. Mode Destination
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			18 05 xb hh ll
			18 02 xb xb
			18 02 xb xb ff
			18 02 xb xb ee ff
			18 02 xb xb
			18 02 xb xb ee ff
				RPPW
				RPOW
				RPPW
				RPOPW
				RPIOW
				RPPIOW
					NA
					NA
					NA
					NA
					NA
					NA
						----	----

MOVW oprx9_xysp, opr16a[1]
MOVW oprx9_xysp, oprx0_xysp[1]
MOVW oprx9_xysp, oprx9_xysp[1]
MOVW oprx9_xysp, oprx16_xysp[1]
MOVW oprx9_xysp, [D_xysp][1]
MOVW oprx9_xysp, [oprx16_xysp][1]
	$(M{:}M+1_1) \Rightarrow M{:}M+1_2$
	Memory to Memory Word-Move (16-Bit),
	IDX1 Source fi Addr. Mode Destination
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			18 05 xb ff hh ll
			18 02 xb ff xb
			18 02 xb ff xb ff
			18 02 xb ff xb ee ff
			18 02 xb ff xb
			18 02 xb ff xb ee ff
				PROPW
				PROOW
				PROPW
				PROOPW
				PROIOW
				PROPIOW
					NA
					NA
					NA
					NA
					NA
					NA
						----	----

MOVW oprx16_xysp, opr16a[1]
MOVW oprx16_xysp, oprx0_xysp[1]
MOVW oprx16_xysp, oprx9_xysp[1]
MOVW oprx16_xysp, oprx16_xysp[1]
MOVW oprx16_xysp, [D_xysp][1]
MOVW oprx16_xysp, [oprx16_xysp][1]
	$(M{:}M+1_1) \Rightarrow M{:}M+1_2$
	Memory to Memory Word-Move (16-Bit),
	IDX2 Source fi Addr. Mode Destination
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			18 05 xb ee ff hh ll
			18 02 xb ee ff xb
			18 02 xb ee ff xb ff
			18 02 xb ee ff xb ee ff
			18 02 xb ee ff xb
			18 02 xb ee ff xb ee ff
				PRPPW
				PRPOW
				
				PRPPW
				
				PRPOPW
				
				PRPIOW
				
				PRPPIO
					NA
					NA
					NA
					NA
					NA
					NA
						----	----

Notes: 1. The first operand in the source code statement specifies the source for the move.
2. The IDX destination code is listed before the source for backwards compatibility.

Table A-1. Instruction Set Summary (Sheet 14 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
MOVW [D_xysp], opr16a[1] **MOVW [D_xysp], oprx0_xysp[1]** **MOVW [D_xysp], oprx9_xysp[1]** **MOVW [D_xysp], oprx16_xysp[1]** **MOVW [D_xysp], [D_xysp][1]** **MOVW [D_xysp], [oprx16_xysp][1]**	**(M:M+1$_1$) ⇒ M:M+1$_2$** **Memory to Memory Word-Move (16-Bit),** **[D,IDX] Source fi Addr. Mode Destination**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 05 xb hh ll** **18 02 xb xb** **18 02 xb xb ff** **18 02 xb xb ee ff** **18 02 xb xb** **18 02 xb xb ee ff**	**IPRfPW** **IPRfOW** **IPRfPW** **IPRfOPW** **IPRfIOW** **IPRfPIOW**	**NA** **NA** **NA** **NA** **NA** **NA**	----	----
MOVW [oprx16_xysp], opr16a[1] **MOVW [oprx16_xysp], oprx0_xysp[1]** **MOVW [oprx16_xysp], oprx9_xysp[1]** **MOVW [oprx16_xysp], oprx16_xysp[1]** **MOVW [oprx16_xysp], [D_xysp][1]** **MOVW [oprx16_xysp], [oprx16_xysp][1]**	**(M:M+1$_1$) ⇒ M:M+1$_2$** **Memory to Memory Word-Move (16-Bit),** **[IDX2] Source fi Addr. Mode Destination**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 05 xb ee ff hh ll** **18 02 xb ee ff xb** **18 02 xb ee ff xb ff** **18 02 xb ee ff xb ee ff** **18 02 xb ee ff xb** **18 02 xb ee ff xb ee ff**	**PIPRfPW** **PIPRfOW** **PIPRfPW** **PIPRfOPW** **PIPRfIOW** **PIPRfPIOW**	**NA** **NA** **NA** **NA** **NA** **NA**	----	----
MUL	(A) × (B) ⇒ A:B 8 by 8 Unsigned Multiply	INH	12	O	O	----	– – – Δ
NEG opr16a NEG oprx0_xysp NEG oprx9,xysp NEG oprx16,xysp NEG [D,xysp] NEG [oprx16,xysp]	0 – (M) ⇒ M *equivalent to* ($\overline{M}$) + 1 ⇒ M Two's Complement Negate	EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	70 hh ll 60 xb 60 xb ff 60 xb ee ff 60 xb 60 xb ee ff	rPwO rPw rPwO frPwP fIfrPw fIPrPw	rPwO rPw rPwO frPwP fIfrPw fIPrPw	----	Δ Δ Δ Δ
NEGA	0 – (A) ⇒ A *equivalent to* ($\overline{A}$) + 1 ⇒ A Negate Accumulator A	INH	40	O	O		
NEGB	0 – (B) ⇒ B *equivalent to* ($\overline{B}$) + 1 ⇒ B Negate Accumulator B	INH	50	O	O		
NEGW opr16a **NEGW oprx0_xysp** **NEGW oprx9,xysp** **NEGW oprx16,xysp** **NEGW [D,xysp]** **NEGW [oprx16,xysp]**	**0–(M:M+1)⇒M:M+1 equivalent to ($\overline{M:M+1}$) +1⇒M:M+1** **Two's Complement Negate**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 70 hh ll** **18 60 xb** **18 60 xb ff** **18 60 xb ee ff** **18 60 xb** **18 60 xb ee ff**	**ORPWO** **ORPW** **ORPWO** **OfRPWP** **OfIfRPW** **OfIPRPW**	**NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ
NEGX	**0 – (X) ⇒ X equivalent to ($\overline{X}$) + 1 ⇒ X**	**INH**	**18 40**	**OO**	**NA**		
NEGY	**Negate Index Register X** **0 – (Y) ⇒ Yequivalent to ($\overline{Y}$) + 1 ⇒ Y** **Negate Index Register Y**	**INH**	**18 50**	**OO**	**NA**		
NOP	No Operation	INH	A7	O	O	----	----
ORAA #opr8i ORAA opr8a ORAA opr16a ORAA oprx0_xysp ORAA oprx9,xysp ORAA oprx16,xysp ORAA [D,xysp] ORAA [oprx16,xysp]	(A) ! (M) ⇒ A Logical OR A with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	8A ii 9A dd BA hh ll AA xb AA xb ff AA xb ee ff AA xb AA xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ 0 –
ORAB #opr8i ORAB opr8a ORAB opr16a ORAB oprx0_xysp ORAB oprx9,xysp ORAB oprx16,xysp ORAB [D,xysp] ORAB [oprx16,xysp]	(B) ! (M) ⇒ B Logical OR B with Memory	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	CA ii DA dd FA hh ll EA xb EA xb ff EA xb ee ff EA xb EA xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ 0 –
ORCC #opr8i	(CCR) ! M ⇒ CCR Logical OR CCR with Memory	IMM	14 ii	P	P	⇑ – ⇑ ⇑	⇑ ⇑ ⇑ ⇑
ORX #opr16i **ORX opr8a** **ORX opr16a** **ORX oprx0_xysp** **ORX oprx9,xysp** **ORX oprx16,xysp** **ORX [D,xysp]** **ORX [oprx16,xysp]**	**(X) ! (M:M+1) ⇒ X** **Logical OR X with Memory**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 8A jj kk** **18 9A dd** **18 BA hh ll** **18 AA xb** **18 AA xb ff** **18 AA xb ee ff** **18 AA xb** **18 AA xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ 0 –

Notes: 1. The first operand in the source code statement specifies the source for the move.
2. The IDX destination code is listed before the source for backwards compatibility.

Table A-1. Instruction Set Summary (Sheet 15 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
ORY #opr16i **ORY opr8a** **ORY opr16a** **ORY oprx0_xysp** **ORY oprx9,xysp** **ORY oprx16,xysp** **ORY [D,xysp]** **ORY [oprx16,xysp]**	**(Y) | (M:M+1) ⇒ Y** **Logical OR Y with Memory**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 CA jj kk** **18 DA dd** **18 FA hh ll** **18 EA xb** **18 EA xb ff** **18 EA xb ee ff** **18 EA xb** **18 EA xb ee ff**	**OP** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ 0 –
PSHA	(SP) – 1 ⇒ SP; (A) ⇒ $M_{(SP)}$ Push Accumulator A onto Stack	INH	36	Os	Os	----	----
PSHB	(SP) – 1 ⇒ SP; (B) ⇒ $M_{(SP)}$ Push Accumulator B onto Stack	INH	37	Os	Os	----	----
PSHC	(SP) – 1 ⇒ SP; (CCR) ⇒ $M_{(SP)}$ Push CCR onto Stack	INH	39	Os	Os	----	----
PSHCW	**(SP) – 2 ⇒ SP; (CCR_H:CCR_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$** **Push CCR onto Stack**	**INH**	**18 39**	**OOS**	**NA**	----	----
PSHD	(SP) – 2 ⇒ SP; (A:B) ⇒ $M_{(SP)}$:$M_{(SP+1)}$ Push D Accumulator onto Stack	INH	3B	OS	OS	----	----
PSHX	(SP) – 2 ⇒ SP; (X_H:X_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$ Push Index Register X onto Stack	INH	34	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	----	----
PULA	($M_{(SP)}$) ⇒ A; (SP) + 1 ⇒ SP Pull Accumulator A from Stack	INH	32	ufO	ufO	----	----
PULB	($M_{(SP)}$) ⇒ B; (SP) + 1 ⇒ SP Pull Accumulator B from Stack	INH	33	ufO	ufO	----	----
PULC	($M_{(SP)}$) ⇒ CCR; (SP) + 1 ⇒ SP Pull CCR from Stack	INH	38	ufO	ufO	Δ fl Δ Δ	Δ Δ Δ Δ
PULCW	**($M_{(SP)}$:$M_{(SP+1)}$) ⇒ CCR_H:CCR_L; (SP) + 2 ⇒ SP** **Pull CCR from Stack**	**INH**	**18 38**	**OUfO**	**NA**	Δ fl 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	----	----
PULX	($M_{(SP)}$:$M_{(SP+1)}$) ⇒ X_H:X_L; (SP) + 2 ⇒ SP Pull Index Register X from Stack	INH	30	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	----	----
REV	MIN-MAX rule evaluation Find smallest rule input (MIN). Store to rule outputs unless fuzzy output is already larger (MAX). For rule weights see REVW. Each rule input is an 8-bit offset from the base address in Y. Each rule output is an 8-bit offset from the base address in Y. $FE separates rule inputs from rule outputs. $FF terminates the rule list. REV may be interrupted.	Special	18 3A	Orf(t,tx)O (exit + re-entry replaces comma above if interrupted) ff + Orf(t,	Orf(t,tx)O ff + Orf(t,	--?–	??Δ?
REVW	MIN-MAX rule evaluation Find smallest rule input (MIN), Store to rule outputs unless fuzzy output is already larger (MAX). Rule weights supported, optional. Each rule input is the 16-bit address of a fuzzy input. Each rule output is the 16-bit address of a fuzzy output. The value $FFFE separates rule inputs from rule outputs. $FFFF terminates the rule list. REVW may be interrupted.	Special	18 3B	ORf(t,Tx)O (loop to read weight if enabled) (r,RfRf) (exit + re-entry replaces comma above if interrupted) ffff + ORf(t,	ORf(t,Tx)O (r,RfRf) ffff + ORf(t,	--?–	??Δ!

Table A-1. Instruction Set Summary (Sheet 16 of 20)

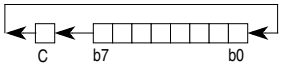
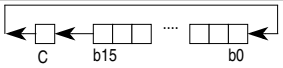
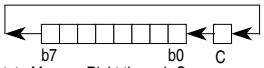
Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
ROL oprx16a ROL oprx0_xysp ROL oprx9,xysp ROL oprx16,xysp ROL [D,xysp] ROL [oprx16,xysp] ROLA ROLB	 Rotate Memory Left through Carry Rotate A Left through Carry Rotate B Left through Carry	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	75 hh ll 65 xb 65 xb ff 65 xb ee ff 65 xb 65 xb ee ff 45 55	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	----	Δ Δ Δ Δ
ROLW oprx16a **ROLW oprx0_xysp** **ROLW oprx9,xysp** **ROLW oprx16,xysp** **ROLW [D,xysp]** **ROLW [oprx16,xysp]** **ROLX** **ROLY**	 **Rotate Memory Left through Carry** **Rotate XLeft through Carry** **Rotate YLeft through Carry**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]** **INH** **INH**	**18 75 hh ll** **18 65 xb** **18 65 xb ff** **18 65 xb ee ff** **18 65 xb** **18 65 xb ee ff** **18 45** **18 55**	**ORPWO** **ORPW** **ORPWO** **OfRPWP** **OfIfRPW** **fOIPRPW** **OO** **OO**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ
ROR opr16a ROR oprx0_xysp ROR oprx9,xysp ROR oprx16,xysp ROR [D,xysp] ROR [oprx16,xysp] RORA RORB	 Rotate Memory Right through Carry Rotate A Right through Carry Rotate B Right through Carry	EXT IDX IDX1 IDX2 [D,IDX] [IDX2] INH INH	76 hh ll 66 xb 66 xb ff 66 xb ee ff 66 xb 66 xb ee ff 46 56	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	rPwO rPw rPwO frPwP fIfrPw fIPrPw O O	----	Δ Δ Δ Δ
RORW oprx16a **RORW oprx0_xysp** **RORW oprx9,xysp** **RORW oprx16,xysp** **RORW [D,xysp]** **RORW [oprx16,xysp]** **RORX** **RORY**	 **Rotate Memory Right through Carry** **Rotate X Right through Carry** **Rotate Y Right through Carry**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]** **INH** **INH**	**18 76 hh ll** **18 66 xb** **18 66 xb ff** **18 66 xb ee ff** **18 66 xb** **18 66 xb ee ff** **18 46** **18 56**	**ORPWO** **ORPW** **ORPWO** **OfRPWP** **OfIfRPW** **OfIPRPW** **OO** **OO**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ
RTC	$(M_{(SP)}) \Rightarrow$ PPAGE; (SP) + 1 ⇒ SP; $(M_{(SP)}:M_{(SP+1)}) \Rightarrow PC_H:PC_L$; (SP) + 2 ⇒ SP Return from Call	INH	0A	uUnfPPP	uUnfPPP	----	----
RTI	$(M_{(SP)}) \Rightarrow$ CCR; (SP) + 1 ⇒ SP $(M_{(SP)}:M_{(SP+1)}) \Rightarrow$ B:A; (SP) + 2 ⇒ SP $(M_{(SP)}:M_{(SP+1)}) \Rightarrow X_H:X_L$; (SP) + 4 ⇒ SP $(M_{(SP)}:M_{(SP+1)}) \Rightarrow PC_H:PC_L$; (SP) – 2 ⇒ SP $(M_{(SP)}:M_{(SP+1)}) \Rightarrow Y_H:Y_L$; (SP) + 4 ⇒ SP Return from Interrupt	INH	0B	UUUUUPPP (with interrupt pending) UUUUUVfPPP	UUUUUPPP UUUUUVfPPP	Δ fl Δ Δ	Δ Δ Δ Δ
RTS	$(M_{(SP)}:M_{(SP+1)}) \Rightarrow PC_H:PC_L$; (SP) + 2 ⇒ SP Return from Subroutine	INH	3D	UfPPP	UfPPP	----	----
SBA	(A) – (B) ⇒ A Subtract B from A	INH	18 16	OO	OO	----	Δ Δ Δ Δ
SBCA #opr8i SBCA opr8a SBCA opr16a SBCA oprx0_xysp SBCA oprx9,xysp SBCA oprx16,xysp SBCA [D,xysp] SBCA [oprx16,xysp]	(A) – (M) – C ⇒ A Subtract with Borrow from A	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	82 ii 92 dd B2 hh ll A2 xb A2 xb ff A2 xb ee ff A2 xb A2 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ Δ Δ
SBCB #opr8i SBCB opr8a SBCB opr16a SBCB oprx0_xysp SBCB oprx9,xysp SBCB oprx16,xysp SBCB [D,xysp] SBCB [oprx16,xysp]	(B) – (M) – C ⇒ B Subtract with Borrow from B	IMM DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	C2 ii D2 dd F2 hh ll E2 xb E2 xb ff E2 xb ee ff E2 xb E2 xb ee ff	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	P rPf rPO rPf rPO frPP fIfrPf fIPrPf	----	Δ Δ Δ Δ

Table A-1. Instruction Set Summary (Sheet 17 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
SBED #opr16i **SBED opr8a** **SBED opr16a** **SBED oprx0_xysp** **SBED oprx9,xysp** **SBED oprx16,xysp** **SBED [D,xysp]** **SBED [oprx16,xysp]**	**(D) – (M:M+1) – C ⇒ D** **Subtract with Borrow from D**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 83 jj kk** **18 93 dd** **18 B3 hh ll** **18 A3 xb** **18 A3 xb ff** **18 A3 xb ee ff** **18 A3 xb** **18 A3 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ
SBEX #opr16i **SBEX opr8a** **SBEX opr16a** **SBEX oprx0_xysp** **SBEX oprx9,xysp** **SBEX oprx16,xysp** **SBEX [D,xysp]** **SBEX [oprx16,xysp]**	**(X) – (M:M+1) – C ⇒ X** **Subtract with Borrow from X**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 82 jj kk** **18 92 dd** **18 B2 hh ll** **18 A2 xb** **18 A2 xb ff** **18 A2 xb ee ff** **18 A2 xb** **18 A2 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ
SBEY #opr16i **SBEY opr8a** **SBEY opr16a** **SBEY oprx0_xysp** **SBEY oprx9,xysp** **SBEY oprx16,xysp** **SBEY [D,xysp]** **SBEY [oprx16,xysp]**	**(Y) – (M:M+1) – C ⇒ Y** **Subtract with Borrow from Y**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 C2 jj kk** **18 D2 dd** **18 F2 hh ll** **18 E2 xb** **18 E2 xb ff** **18 E2 xb ee ff** **18 E2 xb** **18 E2 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ
SEC	1 ⇒ C *Translates to* ORCC #$01	IMM	14 01	P	P	----	---1
SEI	1 ⇒ I; (inhibit I interrupts) *Translates to* ORCC #$10	IMM	14 10	P	P	---1	----
SEV	1 ⇒ V *Translates to* ORCC #$02	IMM	14 02	P	P	----	--1-
SEX abc,dxys	$00:(r1) ⇒ r2 if r1, bit 7 is 0 *or* $FF:(r1) ⇒ r2 if r1, bit 7 is 1 Sign Extend 8-bit r1 to 16-bit r2 r1 may be A, B, or CCR r2 may be D, X, Y, or SP *Alternate mnemonic for* TFR r1, r2	INH	B7 eb	P	P	----	----
STAA opr8a STAA opr16a STAA oprx0_xysp STAA oprx9,xysp STAA oprx16,xysp STAA [D,xysp] STAA [oprx16,xysp]	(A) ⇒ M Store Accumulator A to Memory	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5A dd 7A hh ll 6A xb 6A xb ff 6A xb ee ff 6A xb 6A xb ee ff	Pw PwO Pw PwO PwP PIfw PIPw	Pw PwO Pw PwO PwP PIfw PIPw	----	Δ Δ 0 -
STAB opr8a STAB opr16a STAB oprx0_xysp STAB oprx9,xysp STAB oprx16,xysp STAB [D,xysp] STAB [oprx16,xysp]	(B) ⇒ M Store Accumulator B to Memory	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5B dd 7B hh ll 6B xb 6B xb ff 6B xb ee ff 6B xb 6B xb ee ff	Pw PwO Pw PwO PwP PIfw PIPw	Pw PwO Pw PwO PwP PIfw PIPw	----	Δ Δ 0 -
STD opr8a STD opr16a STD oprx0_xysp STD oprx9,xysp STD oprx16,xysp STD [D,xysp] STD [oprx16,xysp]	(A) ⇒ M, (B) ⇒ M+1 Store Double Accumulator	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5C dd 7C hh ll 6C xb 6C xb ff 6C xb ee ff 6C xb 6C xb ee ff	PW PWO PW PWO PWP PIfW PIPW	PW PWO PW PWO PWP PIfW PIPW	----	Δ Δ 0 -

Table A-1. Instruction Set Summary (Sheet 18 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
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 fi SP; (CCR) ⇒ $M_{(SP)}$; STOP All Clocks Registers stacked to allow quicker recovery by interrupt. If S control bit = 1, the STOP instruction is disabled and acts like a two-cycle NOP.	INH	18 3E	(entering STOP) OOSSSSSf (exiting STOP) fVfPPP (continue) ff (if STOP disabled) OO	(entering STOP) OOSSSSSf (exiting STOP) fVfPPP (continue) ff (if STOP disabled) OO	----	----
STS oprx8a STS oprx16a STS oprx0_xysp STS oprx9,xysp STS oprx16,xysp STS [D,xysp] STS [oprx16,xysp]	(SP_H:SP_L) ⇒ M:M+1 Store Stack Pointer	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5F dd 7F hh ll 6F xb 6F xb ff 6F xb ee ff 6F xb 6F xb ee ff	PW PWO PW PWO PWP PIfW PIPW	PW PWO PW PWO PWP PIfW PIPW	----	Δ Δ 0 –
STX oprx8a STX oprx16a STX oprx0_xysp STX oprx9,xysp STX oprx16,xysp STX [D,xysp] STX [oprx16,xysp]	(X_H:X_L) ⇒ M:M+1 Store Index Register X	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5E dd 7E hh ll 6E xb 6E xb ff 6E xb ee ff 6E xb 6E xb ee ff	PW PWO PW PWO PWP PIfW PIPW	PW PWO PW PWO PWP PIfW PIPW	----	Δ Δ 0 –
STY oprx8a STY oprx16a STY oprx0_xysp STY oprx9,xysp STY oprx16,xysp STY [D,xysp] STY [oprx16,xysp]	(Y_H:Y_L) ⇒ M:M+1 Store Index Register Y	DIR EXT IDX IDX1 IDX2 [D,IDX] [IDX2]	5D dd 7D hh ll 6D xb 6D xb ff 6D xb ee ff 6D xb 6D xb ee ff	PW PWO PW PWO PWP PIfW PIPW	PW PWO PW PWO PWP PIfW PIPW	----	Δ Δ 0 –

SUBA #oprx8i
SUBA oprx8a
SUBA oprx16a
SUBA oprx0_xysp
SUBA oprx9,xysp
SUBA oprx16,xysp
SUBA [D,xysp]
SUBA [oprx16,xysp]
	(A) – (M) ⇒ A
	Subtract Memory from Accumulator A
		IMM
		DIR
		EXT
		IDX
		IDX1
		IDX2
		[D,IDX]
		[IDX2]
			80 ii
			90 dd
			B0 hh ll
			A0 xb
			A0 xb ff
			A0 xb ee ff
			A0 xb
			A0 xb ee ff
				P
				rPf
				rPO
				rPf
				rPO
				frPP
				fIfrPf
				fIPrPf
					P
					rPf
					rPO
					rPf
					rPO
					frPP
					fIfrPf
					fIPrPf
						----	Δ Δ Δ Δ

SUBB #oprx8i
SUBB oprx8a
SUBB oprx16a
SUBB oprx0_xysp
SUBB oprx9,xysp
SUBB oprx16,xysp
SUBB [D,xysp]
SUBB [oprx16,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
					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 [oprx16,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
					PO
					RPf
					RPO
					RPf
					RPO
					fRPP
					fIfRPf
					fIPRPf
						----	Δ Δ Δ Δ

SUBX #opr16i **SUBX opr8a** **SUBX opr16a** **SUBX oprx0_xysp** **SUBX oprx9,xysp** **SUBX oprx16,xysp** **SUBX [D,xysp]** **SUBX [oprx16,xysp]**	**(X) – (M:M+1) ⇒ X** **Subtract Memory from X**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 80 jj kk** **18 90 dd** **18 B0 hh ll** **18 A0 xb** **18 A0 xb ff** **18 A0 xb ee ff** **18 A0 xb** **18 A0 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	----	Δ Δ Δ Δ

Table A-1. Instruction Set Summary (Sheet 19 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
SUBY #opr16i **SUBY opr8a** **SUBY opr16a** **SUBY oprx0_xysp** **SUBY oprx9,xysp** **SUBY oprx16,xysp** **SUBY [D,xysp]** **SUBY [oprx16,xysp]**	**(Y) – (M:M+1) ⇒ Y** **Subtract Memory from Y**	**IMM** **DIR** **EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]**	**18 C0 jj kk** **18 D0 dd** **18 F0 hh ll** **18 E0 xb** **18 E0 xb ff** **18 E0 xb ee ff** **18 E0 xb** **18 E0 xb ee ff**	**OPO** **ORPf** **ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf**	**NA**	---- ---- ---- ---- ---- ---- ---- ----	Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ Δ
SWI	(SP) – $0002 ⇒ SP; $RTN_H : RTN_L ⇒ (M_{(SP)} : M_{(SP+1)})$ (SP) – $0002 ⇒ SP; $Y_H : Y_L ⇒ (M_{(SP)} : M_{(SP+1)})$ (SP) – $0002 ⇒ SP; $X_H : X_L ⇒ (M_{(SP)} : M_{(SP+1)})$ (SP) – $0002 ⇒ SP; B : A⇒ $(M_{(SP)} : M_{(SP+1)})$ (SP) – $0002 ⇒ SP; $CCR_H : CCR_L ⇒ (M_{(SP)} : M_{(SP+1)})$ 1 ⇒ I; (SWI Vector) ⇒ PC Software Interrupt	INH	3F	VSPSSPSSP*	VSPSSPSSP*	---1 (for Reset) 11–1	---- ----
				VfPPP	VfPPP		
*The CPU12 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 fl Δ Δ	Δ Δ Δ Δ
TBA	(B) ⇒ A Transfer B to A	INH	18 0F	OO	OO	----	Δ Δ 0 –
TBEQ abdxys,rel9	If (cntr) = 0, then Branch; else Continue to next instruction Test Counter and Branch if Zero (cntr = A, B, D, X,Y, or SP)	REL (9-bit)	04 lb rr	PPP (branch) PPO (no branch)	PPP (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	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 lb rr	PPP (branch) PPO (no branch)	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	P	---- or Δ fl Δ Δ	---- Δ Δ Δ Δ
TPA	(CCR) ⇒ A *Translates to* TFR CCR ,A	INH	B7 20	P	P	----	----
TRAP trapnum	(SP) – 2 ⇒ SP; $RTN_H:RTN_L ⇒ M_{(SP)}:M_{(SP+1)}$; (SP) – 2 ⇒ SP; $(Y_H:Y_L) ⇒ M_{(SP)}:M_{(SP+1)}$; (SP) – 2 ⇒ SP; $(X_H:X_L) ⇒ M_{(SP)}:M_{(SP+1)}$; (SP) – 2 ⇒ SP; (B:A) ⇒ $M_{(SP)}:M_{(SP+1)}$; (SP) – 1 ⇒ SP; (CCR) ⇒ $M_{(SP)}$ 1 ⇒ I; (TRAP Vector) ⇒ PC Unimplemented opcode trap	INH	18 tn tn = $30–$39 or $40–$FF	OVSPSSPSSP	OVSPSSPSSP	---1	----
TST oprx16a TST oprx0_xysp TST oprx9,xysp TST oprx16,xysp TST [D,xysp] TST [oprx16,xysp] TSTA TSTB	(M) – 0 Test Memory for Zero or Minus (A) – 0Test A for Zero or Minus (B) – 0Test 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	rPO rPf rPO frPP fIfrPf fIPrPf O O	----	Δ Δ 0 0

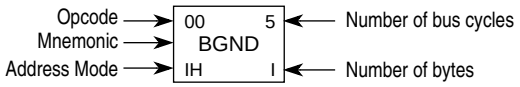
Table A-1. Instruction Set Summary (Sheet 20 of 20)

Source Form	Operation	Addr. Mode	Machine Coding (hex)	Access Detail HCS12X	Access Detail HCS12	S X H I	N Z V C
TSTW oprx16a **TSTW oprx0_xysp** **TSTW oprx9,xysp** **TSTW oprx16,xysp** **TSTW [D,xysp]** **TSTW [oprx16,xysp]** **TSTX** **TSTY**	**(M:M+1) – 0** **Test Memory for Zero or Minus** **(X) – 0Test X for Zero or Minus** **(Y) – 0Test Yfor Zero or Minus**	**EXT** **IDX** **IDX1** **IDX2** **[D,IDX]** **[IDX2]** **INH** **INH**	**18 F7 hh ll** **18 E7 xb** **18 E7 xb ff** **18 E7 xb ee ff** **18 E7 xb** **18 E7 xb ee ff** **18 97** **18 D7**	**ORPO** **ORPf** **ORPO** **OfRPP** **OfIfRPf** **OfIPRPf** **OO** **OO**	**NA** **NA** **NA** **NA** **NA** **NA** **NA** **NA**	---- 	ΔΔ00
TSX	(SP) ⇒ X *Translates to* TFR SP,X	INH	B7 75	P	P	----	----
TSY	(SP) ⇒ Y *Translates to* TFR SP,Y	INH	B7 76	P	P	----	----
TXS	(X) ⇒ SP *Translates to* TFR X,SP	INH	B7 57	P	P	----	----
TYS	(Y) ⇒ SP *Translates to* TFR Y,SP	INH	B7 67	P	P	----	----
WAI	(SP) – 2 ⇒ SP; RTN_H:RTN_L ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (Y_H:Y_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (X_H:X_L) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 2 ⇒ SP; (B:A) ⇒ $M_{(SP)}$:$M_{(SP+1)}$; (SP) – 1 ⇒ SP; (CCR) ⇒ $M_{(SP)}$; WAIT for interrupt	INH	3E	OSSSSsf (after interrupt) fVfPPP	OSSSSsf fVfPPP	---- ---1 -1-1	---- or ---- or ----
WAV	$\sum_{i=1}^{B} S_i F_i$ *fi Y:D* and $\sum_{i=1}^{B} F_i$ *fi 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 Of(frr,ffff)O (add if interrupt) SSS + UUUrr, SSS + UUUrr,		--?-	?Δ??
wavr pseudo-instruction	*see* WAV Resume executing an interrupted WAV instruction (recover intermediate results from stack rather than initializing them to zero)	Special	3C	UUUrr,ffff UUUrr,ffff (frr,ffff)O (frr,ffff)O (exit + re-entry replaces comma above if interrupted) SSS + UUUrr, SSS + UUUrr,		--?-	?Δ??
XGDX	(D) ⇔ (X) *Translates to* EXG D, X	INH	B7 C5	P	P	----	----
XGDY	(D) ⇔ (Y) *Translates to* EXG D, Y	INH	B7 C6	P	P	----	----

Table A-2. Opcode Map (Sheet 1 of 3) — HCS12 and HCS12X Page 1 Opcodes

00 †5	10 1	20 3	30 3	40 1	50 1	60 3-6	70 4	80 1	90 3	A0 3-6	B0 3	C0 1	D0 3	E0 3-6	F0 3
BGND	ANDCC	BRA	PULX	NEGA	NEGB	NEG	NEG	SUBA	SUBA	SUBA	SUBA	SUBB	SUBB	SUBB	SUBB
IH 1	IM 2	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
01 5	11 11	21 1	31 3	41 1	51 1	61 3-6	71 4	81 1	91 3	A1 3-6	B1 3	C1 1	D1 3	E1 3-6	F1 3
MEM	EDIV	BRN	PULY	COMA	COMB	COM	COM	CMPA	CMPA	CMPA	CMPA	CMPB	CMPB	CMPB	CMPB
IH 1	IH 1	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
02 1	12 1	22 3/1	32 3	42 1	52 1	62 3-6	72 4	82 1	92 3	A2 3-6	B2 3	C2 1	D2 3	E2 3-6	F2 3
INY	MUL	BHI	PULA	INCA	INCB	INC	INC	SBCA	SBCA	SBCA	SBCA	SBCB	SBCB	SBCB	SBCB
IH 1	IH 1	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
03 1	13 §1	23 3/1	33 3	43 1	53 1	63 3-6	73 4	83 2	93 3	A3 3-6	B3 3	C3 2	D3 3	E3 3-6	F3 3
DEY	EMUL	BLS	PULB	DECA	DECB	DEC	DEC	SUBD	SUBD	SUBD	SUBD	ADDD	ADDD	ADDD	ADDD
IH 1	IH 1	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3
04 3	14 1	24 3/1	34 2	44 1	54 1	64 3-6	74 4	84 1	94 3	A4 3-6	B4 3	C4 1	D4 3	E4 3-6	F4 3
loop*	ORCC	BCC	PSHX	LSRA	LSRB	LSR	LSR	ANDA	ANDA	ANDA	ANDA	ANDB	ANDB	ANDB	ANDB
RL 3	IM 2	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
05 3-6	15 4-7	25 3/1	35 2	45 1	55 1	65 3-6	75 4	85 1	95 3	A5 3-6	B5 3	C5 1	D5 3	E5 3-6	F5 3
JMP	JSR	BCS	PSHY	ROLA	ROLB	ROL	ROL	BITA	BITA	BITA	BITA	BITB	BITB	BITB	BITB
ID 2-4	ID 2-4	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
06 3	16 4	26 3/1	36 2	46 1	56 1	66 3-6	76 4	86 1	96 3	A6 3-6	B6 3	C6 1	D6 3	E6 3-6	F6 3
JMP	JSR	BNE	PSHA	RORA	RORB	ROR	ROR	LDAA	LDAA	LDAA	LDAA	LDAB	LDAB	LDAB	LDAB
EX 3	EX 3	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
07 4	17 4	27 3/1	37 2	47 1	57 1	67 3-6	77 4	87 1	97 1	A7 1	B7 1	C7 1	D7 1	E7 3-6	F7 3
BSR	JSR	BEQ	PSHB	ASRA	ASRB	ASR	ASR	CLRA	TSTA	NOP	TFR/EXG	CLRB	TSTB	TST	TST
RL 2	DI 2	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IH 1	IH 1	IH 1	IH 2	IH 1	IH 1	ID 2-4	EX 3
08 1	18 -	28 3/1	38 3	48 1	58 1	68 3-6	78 4	88 1	98 3	A8 3-6	B8 3	C8 1	D8 3	E8 3-6	F8 3
INX	Page 2	BVC	PULC	ASLA	ASLB	ASL	ASL	EORA	EORA	EORA	EORA	EORB	EORB	EORB	EORB
IH 1	- -	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
09 1	19 2	29 3/1	39 2	49 1	59 1	69 2-4	79 3	89 1	99 3	A9 3-6	B9 3	C9 1	D9 3	E9 3-6	F9 3
DEX	LEAY	BVS	PSHC	LSRD	ASLD	CLR	CLR	ADCA	ADCA	ADCA	ADCA	ADCB	ADCB	ADCB	ADCB
IH 1	ID 2-4	RL 2	IH 1	IH 1	IH 1	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
0A 7	1A 2	2A 3/1	3A 3	4A 7	5A 2	6A 2-4	7A 3	8A 1	9A 3	AA 3-6	BA 3	CA 1	DA 3	EA 3-6	FA 3
RTC	LEAX	BPL	PULD	CALL	STAA	STAA	STAA	ORAA	ORAA	ORAA	ORAA	ORAB	ORAB	ORAB	ORAB
IH 1	ID 2-4	RL 2	IH 1	EX 4	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
0B †8	1B 2	2B 3/1	3B 2	4B 7-10	5B 2	6B 2-4	7B 3	8B 1	9B 3	AB 3-6	BB 3	CB 1	DB 3	EB 3-6	FB 3
RTI	LEAS	BMI	PSHD	CALL	STAB	STAB	STAB	ADDA	ADDA	ADDA	ADDA	ADDB	ADDB	ADDB	ADDB
IH 1	ID 2-4	RL 2	IH 1	ID 2-5	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3	IM 2	DI 2	ID 2-4	EX 3
0C 4-6	1C 4	2C 3/1	3C +5	4C 4	5C 2	6C 2-4	7C 3	8C 2	9C 3	AC 3-6	BC 3	CC 2	DC 3	EC 3-6	FC 3
BSET	BSET	BGE	wavr	BSET	STD	STD	STD	CPD	CPD	CPD	CPD	LDD	LDD	LDD	LDD
ID 3-5	EX 4	RL 2	SP 1	DI 3	DI 2	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3
0D 4-6	1D 4	2D 3/1	3D 5	4D 4	5D 2	6D 2-4	7D 3	8D 2	9D 3	AD 3-6	BD 3	CD 2	DD 3	ED 3-6	FD 3
BCLR	BCLR	BLT	RTS	BCLR	STY	STY	STY	CPY	CPY	CPY	CPY	LDY	LDY	LDY	LDY
ID 3-5	EX 4	RL 2	IH 1	DI 3	DI 2	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3
0E 4-6	1E 5	2E 3/1	3E †7	4E 4	5E 2	6E 2-4	7E 3	8E 2	9E 3	AE 3-6	BE 3	CE 2	DE 3	EE 3-6	FE 3
BRSET	BRSET	BGT	WAI	BRSET	STX	STX	STX	CPX	CPX	CPX	CPX	LDX	LDX	LDX	LDX
ID 4-6	EX 5	RL 2	IH 1	DI 4	DI 2	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3
0F 4-6	1F 5	2F 3/1	3F 9	4F 4	5F 2	6F 2-4	7F 3	8F 2	9F 3	AF 3-6	BF 3	CF 2	DF 3	EF 3-6	FF 3
BRCLR	BRCLR	BLE	SWI	BRCLR	STS	STS	STS	CPS	CPS	CPS	CPS	LDS	LDS	LDS	LDS
ID 4-6	EX 5	RL 2	IH 1	DI 4	DI 2	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3	IM 3	DI 2	ID 2-4	EX 3

Key to Table A-2



* The opcode $04 (on sheet 1 of 3) corresponds to one of the loop primitive instructions DBEQ, DBNE, IBEQ, IBNE, TBEQ, or TBNE.

Page 2 When the CPU12 encounters a page 2 opcode ($18 on page 1 of the opcode map), it treats the next byte of object code as a page 2 instruction opcode.

† Refer to instruction summary for more information.

§ EMUL requires 3 cycles for HCS12.

Table A-2. Opcode Map (Sheet 2 of 3) — HCS12 Page 2 Opcodes

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

* The opcode $04 (on sheet 1 of 3) corresponds to one of the loop primitive instructions DBEQ, DBNE, IBEQ, IBNE, TBEQ, or TBNE.

† Refer to instruction summary for more information.

Page 2 When the CPU12 encounters a page 2 opcode ($18 on page 1 of the opcode map), it treats the next byte of object code as a page 2 instruction opcode.

Table A-2. Opcode Map (Sheet 3 of 3) — HCS12X Page 2 Opcodes

00 4-6 MOVW IM-ID 5	10 12 IDIV IH 2	20 4 LBRA RL 4	30 10 TRAP IH 2	40 2 NEGX IH 2	50 2 NEGY IH 2	60 4-7 NEGW ID 3-5	70 5 NEGW EX 4	80 3 SUBX IM 4	90 4 SUBX DI 3	A0 4-7 SUBX ID 3-5	B0 4 SUBX EX 4	C0 3 SUBY IM 4	D0 4 SUBY DI 3	E0 4-7 SUBY ID 3-5	F0 4 SUBY EX 4
01 5-7 MOVW EX-ID 5	11 12 FDIV IH 2	21 3 LBRN RL 4	31 10 TRAP IH 2	41 2 COMX IH 2	51 2 COMY IH 2	61 4-7 COMW ID 3-5	71 5 COMW EX 4	81 10 TRAP IH 2	91 10 TRAP IH 2	A1 10 TRAP IH 2	B1 10 TRAP IH 2	C1 10 TRAP IH 2	D1 10 TRAP IH 2	E1 10 TRAP IH 2	F1 10 TRAP IH 2
02 5-10 MOVW ID-ID 4	12 9 EMACS SP 4	22 4/3 LBHI RL 4	32 10 TRAP IH 2	42 2 INCX IH 2	52 2 INCY IH 2	62 4-7 INCW ID 3-5	72 5 INCW EX 4	82 3 SBEX IM 4	92 4 SBEX DI 3	A2 4-7 SBEX ID 3-5	B2 4 SBEX EX 4	C2 3 SBEY IM 4	D2 4 SBEY DI 3	E2 4-7 SBEY ID 3-5	F2 4 SBEY EX 4
03 5 MOVW IM-EX 6	13 3 EMULS IH 2	23 4/3 LBLS RL 4	33 10 TRAP IH 2	43 2 DECX IH 2	53 2 DECY IH 2	63 4-7 DECW ID 3-5	73 5 DECW EX 4	83 3 SBED IM 4	93 4 SBED DI 3	A3 4-7 SBED ID 3-5	B3 4 SBED EX 4	C3 3 ADED IM 4	D3 4 ADED DI 3	E3 4-7 ADED ID 3-5	F3 4 ADED EX 4
04 6 MOVW EX-EX 6	14 12 EDIVS IH 2	24 4/3 LBCC RL 4	34 10 TRAP IH 2	44 2 LSRX IH 2	54 2 LSRY IH 2	64 4-7 LSRW ID 3-5	74 5 LSRW EX 4	84 3 ANDX IM 4	94 4 ANDX DI 3	A4 4-7 ANDX ID 3-5	B4 3 ANDX EX 4	C4 3 ANDY IM 4	D4 4 ANDY DI 3	E4 4-7 ANDY ID 3-5	F4 3 ANDY EX 4
05 5-8 MOVW ID-EX 5	15 12 IDIVS IH 2	25 4/3 LBCS RL 4	35 5 BTAS DI 4	45 2 ROLX IH 2	55 2 ROLY IH 2	65 4-7 ROLW ID 3-5	75 5 ROLW EX 4	85 3 BITX IM 4	95 4 BITX DI 3	A5 4-7 BITX ID 3-5	B5 3 BITX EX 4	C5 3 BITY IM 4	D5 4 BITY DI 3	E5 4-7 BITY ID 3-5	F5 3 BITY EX 4
06 2 ABA IH 2	16 2 SBA IH 2	26 4/3 LBNE RL 4	36 6 BTAS EX 5	46 2 RORX IH 2	56 2 RORY IH 2	66 4-7 RORW ID 3-5	76 5 RORW EX 4	86 10 TRAP IH 2	96 4 GLDAA DI 3	A6 4-7 GLDAA ID 3-5	B6 4 GLDAA EX 4	C6 10 TRAP IH 2	D6 4 GLDAB DI 3	E6 4-7 GLDAB ID 3-5	F6 4 GLDAB EX 4
07 3 DAA IH 2	17 2 CBA IH 2	27 4/3 LBEQ RL 4	37 5-7 BTAS ID 4-6	47 2 ASRX IH 2	57 2 ASRY IH 2	67 4-7 ASRW ID 3-5	77 5 ASRW EX 4	87 2 CLRX IH 2	97 2 TSTX IH 2	A7 10 TRAP IH 2	B7 10 TRAP IH 2	C7 2 CLRY IH 2	D7 2 TSTY IH 2	E7 4-7 TSTW ID 3-5	F7 4 TSTW EX 4
08 4-6 MOVB IM-ID 4	18 4-7 MAXA ID 3-5	28 4/3 LBVC RL 4	38 4 PULCW IH 2	48 2 ASLX IH 2	58 2 ASLY IH 2	68 4-7 ASLW ID 3-5	78 5 ASLW EX 4	88 3 EORX IM 4	98 4 EORX DI 3	A8 4-7 EORX ID 3-5	B8 3 EORX EX 4	C8 3 EORY IM 4	D8 4 EORY DI 3	E8 4-7 EORY ID 3-5	F8 3 EORY EX 4
09 5-7 MOVB EX-ID 5	19 4-7 MINA ID 3-5	29 4/3 LBVS RL 4	39 3 PSHCW IH 2	49 10 TRAP IH 2	59 10 TRAP IH 2	69 4-7 CLRW ID 3-5	79 5 CLRW EX 4	89 3 ADEX IM 4	99 4 ADEX DI 3	A9 4-7 ADEX ID 3-5	B9 4 ADEX EX 4	C9 3 ADEY IM 4	D9 4 ADEY DI 3	E9 4-7 ADEY ID 3-5	F9 4 ADEY EX 4
0A 5-10 MOVB ID-ID 4	1A 4-7 EMAXD ID 3-5	2A 4/3 LBPL RL 4	3A †3n REV SP 2	4A 10 TRAP IH 2	5A 3 GSTAA DI 3	6A 3-5 GSTAA ID 3-5	7A 4 GSTAA EX 4	8A 3 ORX IM 4	9A 4 ORX DI 3	AA 4-7 ORX ID 3-5	BA 3 ORX EX 4	CA 3 ORY IM 4	DA 4 ORY DI 3	EA 4-7 ORY ID 3-5	FA 3 ORY EX 4
0B 4 MOVB IM-EX 5	1B 4-7 EMIND ID 3-5	2B 4/3 LBMI RL 4	3B †5n/3n REVW SP 2	4B 10 TRAP IH 2	5B 3 GSTAB DI 3	6B 3-5 GSTAB ID 3-5	7B 4 GSTAB EX 4	8B 3 ADDX IM 4	9B 4 ADDX DI 3	AB 4-7 ADDX ID 3-5	BB 4 ADDX EX 4	CB 3 ADDY IM 4	DB 4 ADDY DI 3	EB 4-7 ADDY ID 3-5	FB 4 ADDY EX 4
0C 6 MOVB EX-EX 6	1C 4-7 MAXM ID 3-5	2C 4/3 LBGE RL 4	3C †7B WAV SP 2	4C 10 TRAP IH 2	5C 3 GSTD DI 3	6C 3-5 GSTD ID 3-5	7C 4 GSTD EX 4	8C 3 CPED IM 4	9C 4 CPED DI 3	AC 4-7 CPED ID 3-5	BC 4 CPED EX 4	CC 10 TRAP IH 2	DC 4 GLDD DI 3	EC 4-7 GLDD ID 3-5	FC 4 GLDD EX 4
0D 5-8 MOVB ID-EX 5	1 D4-7 MINM ID 3-5	2D 4/3 LBLT RL 4	3D 6 TBL ID 3	4D 10 TRAP IH 2	5D 3 GSTY DI 3	6D 3-5 GSTY ID 3-5	7D 4 GSTY EX 4	8D 3 CPEY IM 4	9D 4 CPEY DI 3	AD 4-7 CPEY ID 3-5	BD 4 CPEY EX 4	CD 10 TRAP IH 2	DD 4 GLDY DI 3	ED 4-7 GLDY ID 3-5	FD 4 GLDY EX 4
0E 2 TAB IH 2	1E 4-7 EMAXM ID 3-5	2E 4/3 LBGT RL 4	3E 8 STOP IH 2	4E 10 TRAP IH 2	5E 3 GSTX DI 3	6E 3-5 GSTX ID 3-5	7E 4 GSTX EX 4	8E 3 CPEX IM 4	9E 4 CPEX DI 3	AE 4-7 CPEX ID 3-5	BE 4 CPEX EX 4	CE 10 TRAP IH 2	DE 4 GLDX DI 3	EE 4-7 GLDX ID 3-5	FE 4 GLDX EX 4
0F 2 TBA IH 2	1F 4-7 EMINM ID 3-5	2F 4/3 LBLE RL 4	3F 8 ETBL ID 3	4F 10 TRAP IH 2	5F 3 GSTS DI 3	6F 3-5 GSTS ID 3-5	7F 4 GSTS EX 4	8F 3 CPES IM 4	9F 4 CPES DI 3	AF 4-7 CPES ID 3-5	BF 4 CPES EX 4	CF 10 TRAP IH 2	DF 4 GLDS DI 3	EF 4-7 GLDS ID 3-5	FF 4 GLDS EX 4

* The opcode $04 (on sheet 1 of 3) corresponds to one of the loop primitive instructions DBEQ, DBNE, IBEQ, IBNE, TBEQ, or TBNE.

† Refer to instruction summary for more information.

Page 2 When the CPU12 encounters a page 2 opcode ($18 on page 1 of the opcode map), it treats the next byte of object code as a page 2 instruction opcode.

Table A-3. 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 indr	F3 [n,SP] 16b indr
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-dec	38 8,X– post-dec	48 8,Y 5b const	58 –8,Y 5b const	68 8,–Y pre-dec	78 8,Y– post-dec	88 8,SP 5b const	98 –8,SP 5b const	A8 8,–SP pre-dec	B8 8,SP– post-dec	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-dec	39 7,X– post-dec	49 9,Y 5b const	59 –7,Y 5b const	69 7,–Y pre-dec	79 7,Y– post-dec	89 9,SP 5b const	99 –7,SP 5b const	A9 7,–SP pre-dec	B9 7,SP– post-dec	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-dec	3A 6,X– post-dec	4A 10,Y 5b const	5A –6,Y 5b const	6A 6,–Y pre-dec	7A 6,Y– post-dec	8A 10,SP 5b const	9A –6,SP 5b const	AA 6,–SP pre-dec	BA 6,SP– post-dec	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-dec	3B 5,X– post-dec	4B 11,Y 5b const	5B –5,Y 5b const	6B 5,–Y pre-dec	7B 5,Y– post-dec	8B 11,SP 5b const	9B –5,SP 5b const	AB 5,–SP pre-dec	BB 5,SP– post-dec	CB 11,PC 5b const	DB –5,PC 5b const	EB [n,Y] 16b indr	FB [n,PC] 16b indr
0C 12,X 5b const	1C –4,X 5b const	2C 4,–X pre-dec	3C 4,X– post-dec	4C 12,Y 5b const	5C –4,Y 5b const	6C 4,–Y pre-dec	7C 4,Y– post-dec	8C 12,SP 5b const	9C –4,SP 5b const	AC 4,–SP pre-dec	BC 4,SP– post-dec	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-dec	3D 3,X– post-dec	4D 13,Y 5b const	5D –3,Y 5b const	6D 3,–Y pre-dec	7D 3,Y– post-dec	8D 13,SP 5b const	9D –3,SP 5b const	AD 3,–SP pre-dec	BD 3,SP– post-dec	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-dec	3E 2,X– post-dec	4E 14,Y 5b const	5E –2,Y 5b const	6E 2,–Y pre-dec	7E 2,Y– post-dec	8E 14,SP 5b const	9E –2,SP 5b const	AE 2,–SP pre-dec	BE 2,SP– post-dec	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-dec	3F 1,X– post-dec	4F 15,Y 5b const	5F –1,Y 5b const	6F 1,–Y pre-dec	7F 1,Y– post-dec	8F 15,SP 5b const	9F –1,SP 5b const	AF 1,–SP pre-dec	BF 1,SP– post-dec	CF 15,PC 5b const	DF –1,PC 5b const	EF [D,Y] D indirect	FF [D,PC] D indirect

Key to Table A-3

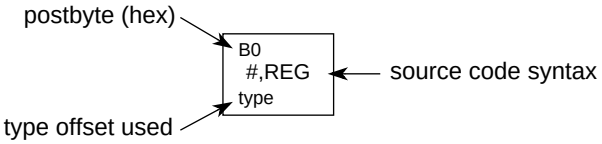


Table A-4. Indexed Addressing Mode Summary

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 A-5. Transfer and Exchange Postbyte Encoding

TRANSFERS									
	MS ⇒	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
⇓ LS		**A**	**B**	**CCR**	**TMPx**	**D**	**X**	**Y**	**SP**
0	**A**	A ⇒ A TFR A,A	B ⇒ A TFR B,A	CCR_L ⇒ A TFR CCR,A TFR CCRL,A	$TMP3_L$ ⇒ A TFR TMP3,A TFR TMP3L,A	B ⇒ A TFR D,A	X_L ⇒ A TFR X, A TFR XL,A	Y_L ⇒ A TFR Y,A TFR YL,A	SP_L ⇒ A TFR SP,A TFR SPL,A
1	**B**	A ⇒ B TFR A,B	B ⇒ B TFR B,B	CCR_L ⇒ B TFR CCR,B TFR CCRL,B	$TMP3_L$ ⇒ B TFR TMP3,B TFR TMP3L,B	B ⇒ B TFR D,B	X_L ⇒ B TFR X, B TFR XL,B	Y_L ⇒ B TFR Y,B TFR YL,B	SP_L ⇒ B TFR SP,B TFR SPL,B
2	**CCR**	A ⇒ CCR TFR A,CCR TFR A,CCRL	B ⇒ CCR TFR B,CCR TFR B,CCRL	CCR_L ⇒ CCR_L TFR CCR,CCR TFR CCRL,CCRL	$TMP3_L$ ⇒ CCR TFR TMP3,CCR TFR TMP3L,CCRL	B ⇒ CCR TFR D,CCR TFR D,CCRL	X_L ⇒ CCR TFR X,CCR TFR XL,CCRL	Y_L ⇒ CCR TFR Y,CCR TFR YL,CCRL	SP_L ⇒ CCR TFR SP,CCR TFR SPL,CCRL
3	**TMP2**	sex:A ⇒ TMP2 SEX A,TMP2	sex:B ⇒ TMP2 SEX B,TMP2	sex:CCR_L ⇒ TMP2 SEX CCR,TMP2 SEX CCRL,TMP2	TMP3 ⇒ TMP2 TFR TMP3,TMP2	D ⇒ TMP2 TFR D,TMP2	X ⇒ TMP2 TFR X,TMP2	Y ⇒ TMP2 TFR Y,TMP2	SP ⇒ TMP2 TFR SP,TMP2
4	**D**	sex:A ⇒ D SEX A,D	sex:B ⇒ D SEX B,D	sex:CCR_L ⇒ D SEX CCR_L,D SEX CCRL,D	TMP3 ⇒ D TFR TMP3,D	D ⇒ D TFR D,D	X ⇒ D TFR X,D	Y ⇒ D TFR Y,D	SP ⇒ D TFR SP,D
5	**X**	sex:A ⇒ X SEX A,X	sex:B ⇒ X SEX B,X	sex:CCR_L ⇒ X SEX CCR,X SEX CCRL,X	TMP3 ⇒ X TFR TMP3,X	D ⇒ X TFR D,X	X ⇒ X TFR X,X	Y ⇒ X TFR Y,X	SP ⇒ X TFR SP,X
6	**Y**	sex:A ⇒ Y SEX A,Y	sex:B ⇒ Y SEX B,Y	sex:CCR_L ⇒ Y SEX CCR,Y SEX CCRL,Y	TMP3 ⇒ Y TFR TMP3,Y	D ⇒ Y TFR D,Y	X ⇒ Y TFR X,Y	Y ⇒ Y TFR Y,Y	SP ⇒ Y TFR SP,Y
7	**SP**	sex:A ⇒ SP SEX A,SP	sex:B ⇒ SP SEX B,SP	sex:CCR_L ⇒ SP SEX CCR,SP SEX CCRL,SP	TMP3 ⇒ SP TFR TMP3,SP	D ⇒ SP TFR D,SP	X ⇒ SP TFR X,SP	Y ⇒ SP TFR Y,SP	SP ⇒ SP TFR SP,SP
8	**A**	A ⇒ A TFR A,A	B ⇒ A TFR B,A	CCR_H ⇒ A TFR CCRH,A	$TMP3_H$ ⇒ A TFR TMP3H,A	B ⇒ A TFR D,A	X_H ⇒ A TFR XH, A	Y_H ⇒ A TFR YH,A	SP_H ⇒ A TFR SPH,A
9	**B**	A ⇒ B TFR A,B	B ⇒ B TFR B,B	CCR_L ⇒ B TFR CCRL,B	$TMP3_L$ ⇒ B TFR TMP3L,B	B ⇒ B TFR D,B	X_L ⇒ B TFR XL, B	Y_L ⇒ B TFR YL,B	SP_L ⇒ B TFR SPL,B
A	**CCR**	A ⇒ CCR_H TFR A,CCRH	B ⇒ CCR_L TFR B,CCRL	CCRW ⇒ CCRW TFR CCRW,CCRW	TMP3 ⇒ $CCR_{H:L}$ TFR TMP3,CCRW	D ⇒ $CCR_{H:L}$ TFR D,CCRW	X ⇒ $CCR_{H:L}$ TFR X,CCRW	Y ⇒ $CCR_{H:L}$ TFR Y,CCRW	SP ⇒ $CCR_{H:L}$ TFR SP,CCRW
B	**TMPx**	A ⇒ $TMP2_H$ TFR A,TMP2H	B ⇒ $TMP2_L$ TFR B,TMP2L	$CCR_{H:L}$ ⇒ TMP2 TFR CCRW,TMP2	TMP3 ⇒ TMP2 TFR TMP3,TMP2	D ⇒ TMP1 TFR D,TMP1	X ⇒ TMP2 TFR X,TMP2	Y ⇒ TMP2 TFR Y,TMP2	SP ⇒ TMP2 TFR SP,TMP2
C	**D**	sex:A ⇒ D SEX A,D	sex:B ⇒ D SEX B,D	$CCR_{H:L}$ ⇒ D TFR CCRW,D	TMP1 ⇒ D TFR TMP1,D	D ⇒ D TFR D,D	X ⇒ D TFR X,D	Y ⇒ D TFR Y,D	SP ⇒ D TFR SP,D
D	**X**	A ⇒ X_H TFR A,XH	B ⇒ X_L TFR B,XL	$CCR_{H:L}$ ⇒ X TFR CCRW,X	TMP3 ⇒ X TFR TMP3,X	sex:D ⇒ X SEX D,X	X ⇒ X TFR X,X	Y ⇒ X TFR Y,X	SP ⇒ X TFR SP,X
E	**Y**	A ⇒ Y_H TFR A,YH	B ⇒ Y_L TFR B,YL	$CCR_{H:L}$ ⇒ Y TFR CCRW,Y	TMP3 ⇒ Y TFR TMP3,Y	sex:D ⇒ Y SEX D,Y	X ⇒ Y TFR X,Y	Y ⇒ Y TFR Y,Y	SP ⇒ Y TFR SP,Y
F	**SP**	A ⇒ SP_H TFR A,SPH	B ⇒ SP_L TFR B,SPL	$CCR_{H:L}$ ⇒ SP TFR CCRW,SP	TMP3 ⇒ SP TFR TMP3,SP	D ⇒ SP TFR D,SP	X ⇒ SP TFR X,SP	Y ⇒ SP TFR Y,SP	SP ⇒ SP TFR SP,SP

Note: Encodings in the shaded area (LS = 8–F) are only available on the S12X.

Table A-5. Transfer and Exchange Postbyte Encoding (continued)

EXCHANGES									
	MS⇒	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
⇓ LS		**A**	**B**	**CCR**	**TMPx**	**D**	**X**	**Y**	**SP**
0	**A**	A ⇔ A EXG A,A	B ⇔ A EXG B,A	CCR_L ⇔ A EXG CCR,A *EXG CCRL,A*	$TMP3_L$ ⇒ A $00:A ⇒ TMP3 EXG A, TMP3	B ⇔ A EXG D,A	X_L ⇒ A $00:A ⇒ X EXG X,A	Y_L ⇒ A $00:A ⇒ Y EXG Y,A	SP_L ⇒ A $00:A ⇒ SP EXG SP,A
1	**B**	A ⇔ B EXG A,B	B ⇔ B EXG B,B	CCR_L ⇔ B EXG CCR,B *EXG CCRL,B*	$TMP3_L$ ⇒ B $FF:B ⇒ TMP3 EXG B,TMP3	B ⇒ B $FF ⇒ A EXG D,B	X_L ⇒ B $FF:B ⇒ X EXG X,B	Y_L ⇒ B $FF:B ⇒ Y EXG Y,B	SP_L ⇒ B $FF:B ⇒ SP EXG SP,B
2	**CCR**	A ⇔ CCR_L EXG A, CCR *EXG A,CCRL*	B ⇔ CCR_L EXG B,CCR *EXG B,CCRL*	CCR_L ⇔ CCR_L EXG CCR,CCR *EXG CCRL,CCRL*	$TMP3_L$ ⇒ CCR_L $FF:CCR$_L$ ⇒ TMP3 EXG, TMP3,CCR *EXG TMP3,CCRL*	B ⇒ CCR_L $FF:CCR$_L$ ⇒ D EXG D,CCR *EXG D,CCRL*	X_L ⇒ CCR_L $FF:CCR$_L$ ⇒ X EXG X,CCR *EXG X,CCRL*	Y_L ⇒ CCR_L $FF:CCR$_L$ ⇒ Y EXG Y,CCR *EXG Y,CCRL*	SP_L ⇒ CCR_L $FF:CCR$_L$ ⇒ SP EXG SP,CCR *EXG SP,CCRL*
3	**TMP2**	$00:A ⇒ TMP2 $TMP2_L$ ⇒ A EXG A,TMP2	$00:B ⇒ TMP2 $TMP2_L$ ⇒ B EXG B,TMP2	$00:CCR$_L$ ⇒ TMP2 $TMP2_L$ ⇒ CCR EXG CCR,TMP2	TMP3 ⇔ TMP2 EXG TMP3,TMP2	D ⇔ TMP2 EXG D,TMP2	X ⇔ TMP2 EXG X,TMP2	Y ⇔ TMP2 EXG Y,TMP2	SP ⇔ TMP2 EXG SP,TMP2
4	**D**	$00:A ⇒ D EXG A,D	$00:B ⇒ D EXG B,D	$00:CCR$_L$ ⇒ D B ⇒ CCR_L EXG CCR,D *EXG CCRL,D*	TMP3 ⇔ D EXG TMP3,D	D ⇔ D EXG D,D	X ⇔ D EXG X,D	Y ⇔ D EXG Y,D	SP ⇔ D EXG SP,D
5	**X**	$00:A ⇒ X X_L ⇒ A EXG A,X	$00:B ⇒ X X_L ⇒ B EXG B,X	$00:CCR$_L$ ⇒ X X_L ⇒ CCR_L EXG CCR,X *EXG CCRL,X*	TMP3 ⇔ X EXG TMP3,X	D ⇔ X EXG D,X	X ⇔ X EXG X,X	Y ⇔ X EXG Y,X	SP ⇔ X EXG SP,X
6	**Y**	$00:A ⇒ Y Y_L ⇒ A EXG A,Y	$00:B ⇒ Y Y_L ⇒ B EXG B,Y	$00:CCR$_L$ ⇒ Y Y_L ⇒ CCR_L EXG CCR,X *EXG CCRL,X*	TMP3 ⇔ Y EXG TMP3,Y	D ⇔ Y EXG D,Y	X ⇔ Y EXG X,Y	Y ⇔ Y EXG Y,Y	SP ⇔ Y EXG SP,Y
7	**SP**	$00:A ⇒ SP SP_L ⇒ A EXG A,SP	$00:B ⇒ SP SP_L ⇒ B EXG B,SP	$00:CCR$_L$ ⇒ SP SP_L ⇒ CCR_L EXG CCR,X *EXG CCRL,X*	TMP3 ⇔ SP EXG TMP3,SP	D ⇔ SP EXG D,SP	X ⇔ SP EXG X,SP	Y ⇔ SP EXG Y,SP	SP ⇔ SP EXG SP,SP
8	**A**	A ⇔ A EXG A,A	B ⇔ A EXG B,A	CCR_H ⇔ A EXG CCRH,A	$TMP3_H$ ⇔ A EXG TMP3H,A	B ⇔ A EXG D,A	X_H ⇔ A EXG XH,A	Y_H ⇔ A EXG YH,A	SP_H ⇔ A EXG SPH,A
9	**B**	A ⇔ B EXG A,B	B ⇔ B EXG B,B	CCR_L ⇔ B EXG CCRL,B	$TMP3_L$ ⇔ B EXG TMP3L,B	$FF ⇒ A, B ⇒ B EXG D,B	X_L ⇔ B EXG XL,B	Y_L ⇔ B EXG YL,B	SP_L ⇔ B EXG SPL,B
A	**CCR**	A ⇔ CCR_H EXG A,CCRH	B ⇔ CCR_L EXG B,CCRL	$CCR_{H:L}$ ⇔ $CCR_{H:L}$ EXG CCRW,CCRW	TMP3 ⇔ $CCR_{H:L}$ EXG TMP3,CCRW	D ⇔ $CCR_{H:L}$ EXG D,CCRW	X ⇔ $CCR_{H:L}$ EXG X,CCRW	Y ⇔ $CCR_{H:L}$ EXG Y,CCRW	SP ⇔ $CCR_{H:L}$ EXG, SP,CCRW
B	**TMPx**	A ⇔ $TMP2_H$ EXG A,TMP2H	B ⇔ $TMP2_L$ EXG B,TMP2L	$CCR_{H:L}$⇔ TMP2 EXG CCRW,TMP2	TMP3 ⇔ TMP2 EXG TMP3,TMP2	D ⇔ TMP1 EXG D,TMP1	X ⇔ TMP2 EXG X,TMP2	Y ⇔ TMP2 EXG Y,TMP2	SP ⇔ TMP2 EXG SP,TMP2
C	**D**	$00:A ⇒ D EXG A,D	$00:B ⇒ D EXG B,D	$CCR_{H:L}$ ⇔ D EXG CCRW,D	TMP1 ⇔ D EXG TMP1,D	D ⇔ D EXG D,D	X ⇔ D EXG X,D	Y ⇔ D EXG Y,D	SP ⇔ D EXG SP,D
D	**X**	A ⇔ X_H EXG A,XH	B ⇔ X_L EXG B,XL	$CCR_{H:L}$ ⇔ X EXG CCRW,X	TMP3 ⇔ X EXG TMP3,X	D ⇔ X EXG D,X	X ⇔ X EXG X,X	Y ⇔ X EXG Y,X	SP ⇔ X EXG SP,X
E	**Y**	A ⇔ Y_H EXG A,YH	B ⇔ Y_L EXG B,YL	$CCR_{H:L}$ ⇔ Y EXG CCRW,Y	TMP3 ⇔ Y EXG TMP3,Y	D ⇔ Y EXG D,Y	X ⇔ Y EXG X,Y	Y ⇔ Y EXG Y,Y	SP ⇔ Y EXG SP,Y
F	**SP**	A ⇔ SP_H EXG A,SPH	B ⇔ SP_L EXG B,SPL	$CCR_{H:L}$ ⇔ SP EXG CCRW,SP	TMP3 ⇔ SP EXG TMP3,SP	D ⇔ SP EXG D,SP	X ⇔ SP EXG X,SP	Y ⇔ SP EXG Y,SP	SP ⇔ SP EXG SP,SP

Note: Encodings in the shaded area (LS = 8–F) are only available on the S12X.

Table A-6. Loop Primitive Postbyte Encoding (lb)

00 A DBEQ (+)	10 A DBEQ (–)	20 A DBNE (+)	30 A DBNE (–)	40 A TBEQ (+)	50 A TBEQ (–)	60 A TBNE (+)	70 A TBNE (–)	80 A IBEQ (+)	90 A IBEQ (–)	A0 A IBNE (+)	B0 A IBNE (–)
01 B DBEQ (+)	11 B DBEQ (–)	21 B DBNE (+)	31 B DBNE (–)	41 B TBEQ (+)	51 B TBEQ (–)	61 B TBNE (+)	71 B TBNE (–)	81 B IBEQ (+)	91 B IBEQ (–)	A1 B IBNE (+)	B1 B IBNE (–)
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 D DBEQ (+)	14 D DBEQ (–)	24 D DBNE (+)	34 D DBNE (–)	44 D TBEQ (+)	54 D TBEQ (–)	64 D TBNE (+)	74 D TBNE (–)	84 D IBEQ (+)	94 D IBEQ (–)	A4 D IBNE (+)	B4 D IBNE (–)
05 X DBEQ (+)	15 X DBEQ (–)	25 X DBNE (+)	35 X DBNE (–)	45 X TBEQ (+)	55 X TBEQ (–)	65 X TBNE (+)	75 X TBNE (–)	85 X IBEQ (+)	95 X IBEQ (–)	A5 X IBNE (+)	B5 X IBNE (–)
06 Y DBEQ (+)	16 Y DBEQ (–)	26 Y DBNE (+)	36 Y DBNE (–)	46 Y TBEQ (+)	56 Y TBEQ (–)	66 Y TBNE (+)	76 Y TBNE (–)	86 Y IBEQ (+)	96 Y IBEQ (–)	A6 Y IBNE (+)	B6 Y IBNE (–)
07 SP DBEQ (+)	17 SP DBEQ (–)	27 SP DBNE (+)	37 SP DBNE (–)	47 SP TBEQ (+)	57 SP TBEQ (–)	67 SP TBNE (+)	77 SP TBNE (–)	87 SP IBEQ (+)	97 SP IBEQ (–)	A7 SP IBNE (+)	B7 SP IBNE (–)

Key to Table A-6

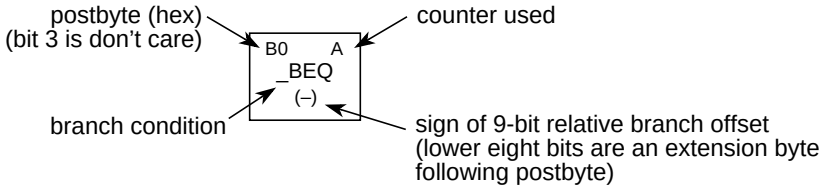


Table A-7. Branch/Complementary Branch

Branch				Complementary Branch			
Test	**Mnemonic**	**Opcode**	**Boolean**	**Test**	**Mnemonic**	**Opcode**	**Comment**
r>m	BGT	2E	Z + (N ⊕ V) = 0	r≤m	BLE	2F	Signed
r≥m	BGE	2C	N ⊕ V = 0	r<m	BLT	2D	Signed
r=m	BEQ	27	Z = 1	r≠m	BNE	26	Signed
r≤m	BLE	2F	Z + (N ⊕ V) = 1	r>m	BGT	2E	Signed
r<m	BLT	2D	N ⊕ V = 1	r≥m	BGE	2C	Signed
r>m	BHI	22	C + Z = 0	r≤m	BLS	23	Unsigned
r≥m	BHS/BCC	24	C = 0	r<m	BLO/BCS	25	Unsigned
r=m	BEQ	27	Z = 1	r≠m	BNE	26	Unsigned
r≤m	BLS	23	C + Z = 1	r>m	BHI	22	Unsigned
r<m	BLO/BCS	25	C = 1	r≥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≠0	BNE	26	Simple
Always	BRA	20	—	Never	BRN	21	Unconditional

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

Table A-8. Hexadecimal to ASCII Conversion

Hex	ASCII	Hex	ASCII	Hex	ASCII	Hex	ASCII
$00	NUL	$20	SP *space*	$40	@	$60	` *grave*
$01	SOH	$21	!	$41	A	$61	a
$02	STX	$22	“ quote	$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 *beep*	$27	‘ *apost*.	$47	G	$67	g
$08	BS *back sp*	$28	(	$48	H	$68	h
$09	HT *tab*	$29	)	$49	I	$69	i
$0A	LF *linefeed*	$2A	*	$4A	J	$6A	j
$0B	VT	$2B	+	$4B	K	$6B	k
$0C	FF	$2C	, *comma*	$4C	L	$6C	l
$0D	CR *return*	$2D	- *dash*	$4D	M	$6D	m
$0E	SO	$2E	. *period*	$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	_ *under*	$7F	DEL *delete*

A.5 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 A-9. The decimal equivalent of the original hexadecimal number is the sum of the weights found in the table for all hexadecimal digits.

Table A-9. 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

A.6 Decimal-to-Hexadecimal Conversion

To convert a decimal number (up to $65{,}535_{10}$) to hexadecimal, find the largest decimal number in Table A-9 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.