# 68HC12 Cycles

- Dragon12-Plus board uses a on 48 MHz clock
  - The on-board oscillator runs at 8 MHz
  - An internal phase-locked loop multiplies this by 6 to get 48 MHz
- A processor cycle takes 2 clock cycles – E clock is 24 MHz
- Each processor cycle takes 41.7 ns (1/24 $\mu$s) to execute
- An instruction takes from 1 to 12 processor cycles to execute
- You can determine how many cycles an instruction takes by looking up the CPU cycles for that instruction in the S12CPUV2 Reference Manual.
  - For example, `LDAA` using the `IMM` addressing mode shows one CPU cycle (of type P).
  - `LDAA` using the `EXT` addressing mode shows three CPU cycles (of type rPO).
  - Section 6.6 of the S12CPUV2 Reference Manual explains what the HCS12 is doing during each of the different types of CPU cycles.

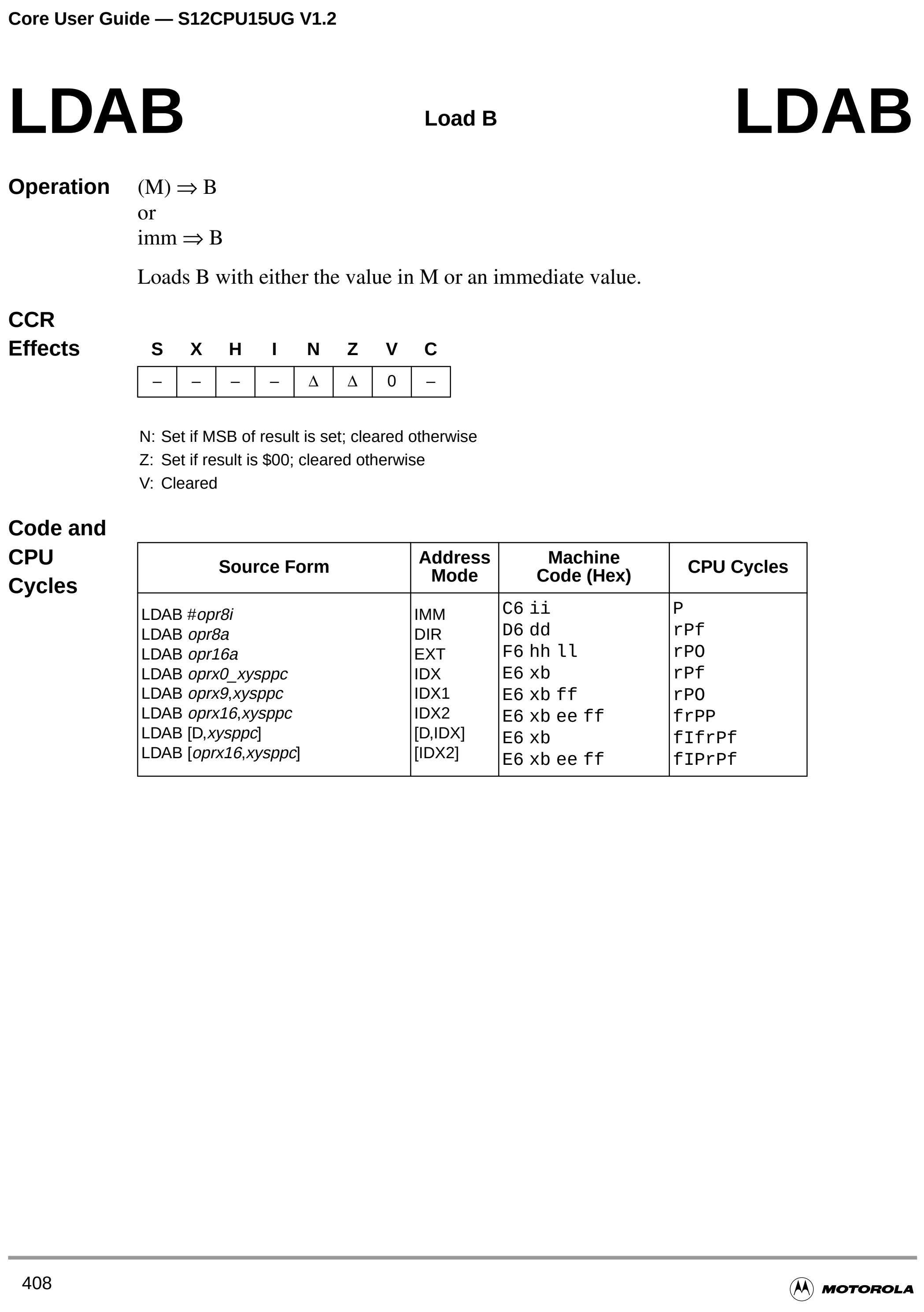
```
000                  org  $2000    ; Inst   Mode   Cycles
2000 C6 0A            ldab #10      ; LDAB   (IMM)    1
2002 87        loop:  clra          ; CLRA   (INH)    1
2003 04 31 FC         dbne b,loop   ; DBNE   (REL)    3
2006 3F               swi           ; SWI             9
```

The program executes the `ldab #10` instruction once (which takes one cycle). It then goes through loop 10 times (which has two instructions, on with one cycle and one with three cycles), and finishes with the `swi` instruction (which takes 9 cycles).

Total number of cycles:

$1 + 10 \times (1 + 3) + 9 = 50$

50 cycles = 50 × 41.7 ns/cycle = 2.08 $\mu$s



# LDAB Load B LDAB

**Operation** (M) ⇒ B
or
imm ⇒ B

Loads B with either the value in M or an immediate value.

**CCR Effects**

| S | X | H | I | N | Z | V | C |
|---|---|---|---|---|---|---|---|
| – | – | – | – | Δ | Δ | 0 | – |

N: Set if MSB of result is set; cleared otherwise
Z: Set if result is $00; cleared otherwise
V: Cleared

**Code and CPU Cycles**

| Source Form | Address Mode | Machine Code (Hex) | CPU Cycles |
|---|---|---|---|
| LDAB #*opr8i* | IMM | C6 ii | P |
| LDAB *opr8a* | DIR | D6 dd | rPf |
| LDAB *opr16a* | EXT | F6 hh ll | rPO |
| LDAB *oprx0_xysppc* | IDX | E6 xb | rPf |
| LDAB *oprx9,xysppc* | IDX1 | E6 xb ff | rPO |
| LDAB *oprx16,xysppc* | IDX2 | E6 xb ee ff | frPP |
| LDAB [D,*xysppc*] | [D,IDX] | E6 xb | fIfrPf |
| LDAB [*oprx16,xysppc*] | [IDX2] | E6 xb ee ff | fIPrPf |

# HC12 Assembly Language Programming

## Programming Model

## Addressing Modes

## Assembler Directives

## HC12 Instructions

## Flow Charts

## Assembler Directives

- In order to write an assembly language program it is necessary to use *assembler directives.*
- These are not instructions which the HC12 executes but are directives to the assembler program about such things as where to put code and data into memory.
- All of the assembler directives can be found in as12.html on the EE 308 home page.
- We will use only a few of these directives. (Note: In the following table, [] means an optional argument.) Here are the ones we will need:

| Directive Name | Description | Example |
|---|---|---|
| **equ** | Give a value to a symbol | `len:   equ    100` |
| **org** | Set starting value of location counter where code or data will go | `       org    $1000` |
| **dc.b**<br>**db**<br>**fcb** | Allocate and initialize storage for 8-bit variables.<br>Place the bytes in successive memory locations | `var:   dc.b   2,18` |
| **dc.w**<br>**dw**<br>**fdb** | Allocate and initialize storage for 16-bit variables.<br>Place the bytes in successive memory locations | `var:   dc.w   $ABCD` |
| **ds.b**<br>**ds**<br>**rmb** | Allocate specified number of 8-bit storage spaces. | `table: ds.w   10` |
| **ds.w**<br>**rmw** | Allocate specified number of 16-bit storage spaces. | `table2: ds.w   50` |
| **fcc** | Encodes a string of ASCII characters. The first characteris the delimiter. The string terminates at the next occurrence of the delimiter | `string: fcc    "Hello"` |
| **fill** | Fill memory with a given value<br>The first value is the number of bytes to fill.<br>The second number is the value to put into memory | `init_data: fill  100,0` |

## Using labels in assembly programs

A **label** is defined by a name followed by a colon as the first thing on a line. When the label is referred to in the program, it has the numerical value of the location counter when the label was defined.

Here is a code fragment using labels and the assembler directives `dc` and `ds`:

```
         org     $2000
table1:  dc.b    $23,$17,$f2,$a3,$56
table2:  ds.b    5
var:     dc.w    $43af
```

The as12 assembler produces a listing file (`.lst`) and a symbol file (`.sym`). Here is the listing file from the assembler:

```
as12, an absolute assembler for Motorola MCU's, version 1.2e

2000                             org     $2000
2000 23 17 f2 a3 56      table1: dc.b    $23,$17,$f2,$a3,$56
2005                     table2: ds.b    5
200A 43 af               var:    dc.w    $43af
```

And here is the symbol file:

```
table1     2000
table2     2005
var        200A
```

Note that `table1` is a name with the value of `$2000`, the value of the location counter defined in the `org` directive. Five bytes of data are defined by the `dc.b` directive, so the location counter is increased from `$2000` to `$2005`. `table2` is a name with the value of `$2005`. Five bytes of data are set aside for `table2` by the `ds.b 5` directive. The as12 assembler initialized these five bytes of data to all zeros. `var` is a name with the value of `$200a`, the first location after `table2`.

# HC12 Assembly Language Programming

Programming Model

Addressing Modes

Assembler Directives

HC12 Instructions

Flow Charts

1. Data Transfer and Manipulation Instructions — instructions which move and manipulate data (**S12CPUV2 Reference Manual**, Sections 5.3, 5.4, and 5.5).

   - Load and Store — load copy of memory contents into a register; store copy of register contents into memory.

     ```
     LDAA  $2000  ; Copy contents of addr $2000 into A
     STD   0,X    ; Copy contents of D to addrs X and X+1
     ```

   - Transfer — copy contents of one register to another.

     ```
     TBA                 ; Copy B to A
     TFR  X,Y            ; Copy X to Y
     ```

   - Exhange — exchange contents of two registers.

     ```
     XGDX                 ; Exchange contents of D and X
     EXG A,B              ; Exchange contents of A and B
     ```

   - Move — copy contents of one memory location to another.

     ```
     MOVB $2000,$20A0    ; Copy byte at $2000 to $20A0
     MOVW 2,X+,2,Y+     ; Copy two bytes from address held
                         ; in X to address held in Y
                         ; Add 2 to X and Y
     ```

2. Arithmetic Instructions — addition, subtraction, multiplication, divison (**S12CPUV2 Reference Manual**, Sections 5.6, 5.8 and 5.12).

   ```
   ABA          ; Add B to A; results in A
   SUBD $20A1   ; Subtract contents of $20A1 from D
   INX          ; Increment X by 1
   MUL          ; Multiply A by B; results in D
   ```

3. Logic and Bit Instructions — perform logical operations (**S12CPUV2 Reference Manual**, Sections 5.9, 5.10, 5.11, 5.13 and 5.14).

   - Logic Instructions

     ```
     ANDA $2000 ; Logical AND of A with contents of $2000
     EORB 2,X   ; Exclusive OR B with contents of address (X+2)
     ```

   - Clear, Complement and Negate Instructions

     ```
     NEG  -2,X  ; Negate (2's comp) contents of address (X-2)
     CLRA       ; Clear Acc A
     ```

- Bit manipulate and test instructions — work with one bit of a register or memory.

  ```
  BITA #$08         ; Check to see if Bit 3 of A is set
  BSET $0002,#$18   ; Set bits 3 and 4 of address $002
  ```

- Shift and rotate instructions

  ```
  LSLA              ; Logical shift left A
  ASR  $1000        ; Arithmetic shift right value at address $1000
  ```

4. Compare and test instructions — test contents of a register or memory (to see if zero, negative, etc.), or compare contents of a register to memory (to see if bigger than, etc.) (**S12CPUV2 Reference Manual**, Section 5.9).

   ```
   TSTA         ; (A)-0 -- set flags accordingly
   CPX  #$8000  ; (X) - $8000 -- set flags accordingly
   ```

5. Jump and Branch Instructions — Change flow of program (e.g., goto, it-then-else, switch-case) (**S12CPUV2 Reference Manual**, Sections 5.19, 5.20 and 5.21).

   ```
   JMP  L1           ; Start executing code at address label L1
   BEQ  L2           ; If Z bit set, go to label L2
   DBNE X,L3         ; Decrement X; if X not 0 then goto L3
   BRCLR $1A,#$80,L4 ; If bit 7 of addr $1A clear, go to label L4
   JSR sub1          ; Jump to subroutine sub1
   RTS               ; Return from subroutine
   ```

6. Interrupt Instructions — Initiate or terminate an interrupt call (**S12CPUV2 Reference Manual**, Section 5.22).

   - Interrupt instructions

     ```
     SWI       ; Initiate software interrupt
     RTI       ; Return from interrupt
     ```

7. Index Manipulation Instructions — Put address into X, Y or SP, manipulate X, Y or SP (**S12CPUV2 Reference Manual**, Section 5.23).

```
ABX        ; Add (B) to (X)
LEAX 5,Y   ; Put address (Y) + 5 into X
```

8. Condition Code Instructions — change bits in Condition Code Register (**S12CPUV2 Reference Manual**, Section 5.26).

```
ANDCC #$f0  ; Clear N, Z, C and V bits of CCR
SEV         ; Set V bit of CCR
```

9. Stacking Instructions — push data onto and pull data off of stack (**S12CPUV2 Reference Manual**, Section 5.24).

```
PSHA       ; Push contents of A onto stack
PULX       ; Pull two top bytes of stack, put into X
```

10. Stop and Wait Instructions — put HC12 into low power mode (**S12CPUV2 Reference Manual**, Section 5.27).

```
STOP       ; Put into lowest power mode
WAI        ; Put into low power mode until next interrupt
```

11. Null Instructions

```
NOP        ; No operation
BRN        ; Branch never
```

12. Instructions we won't discuss or use — BCD arithmetic, fuzzy logic, minimum and maximum, multiply-accumulate, table interpolation (**S12CPUV2 Reference Manual**, Sections 5.7, 5.16, 5.17, and 5.18).

## Disassembly of an HC12 Program

- It is sometimes useful to be able to convert HC12 op codes into mnemonics.
- For example, consider the hex code:

```
ADDR  DATA
---- ---------------------------------------------
1000  C6  05  CE  20  00  E6  01  18  06  04 35 EE 3F
```

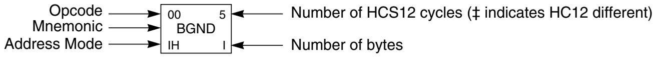
- To determine the instructions, use Tables A.2 through A.7 of the S12CPUV2 Reference Manual.
  - If the first byte of the instruction is anything other than $18, use Sheet 1 of Table A.2 (Page 395). From this table, determine the number of bytes of the instruction and the addressing mode. For example, $C6 is a two-byte instruction, the mnemonic is `LDAB`, and it uses the `IMM` addressing mode. Thus, the two bytes `C6 05` is the op code for the instruction `LDAB #$05`.
  - If the first byte is $18, use Sheet 2 of Table A.2 (Page 396), and do the same thing. For example, `18 06` is a two byte instruction, the mnemonic is `ABA`, and it uses the `INH` addressing mode, so there is no operand. Thus, the two bytes `18 06` is the op code for the instruction `ABA`.
  - Indexed addressing mode is fairly complicated to disassemble. You need to use Table A.3 to determine the operand. For example, the op code `$E6` indicates `LDAB` indexed, and may use two to four bytes (one to three bytes in addition to the op code). The postbyte `01` indicates that the operand is `0,1`, which is 5-bit constant offest, which takes only one additional byte. All 5-bit constant offset, pre and post increment and decrement, and register offset instructions use one additional byte. All 9-bit constant offset instructions use two additional bytes, with the second byte holding 8 bits of the 9 bit offset. (The 9th bit is a direction bit, which is held in the first postbyte.) All 16-bit constant offset instructions use three postbytes, withe the 2nd and 3rd holding the 16-bit unsigned offset.
  - Transfer (`TFR`) and exchange (`EXG`) instructions all have the op code `$B7`. Use Table A.5 to determine whether it is `TFR` or an `EXG`, and to determine which registers are being used. If the most significant bit of the postbyte is 0, the instruction is a transfer instruction.
  - Loop instructions (*Decrement and Branch*, *Increment and Branch*, and *Test and Branch*) all have the op code `$04`. To determine which instruction the op code `$04` implies, and whether the branch is positive (forward) or negative (backward), use Table A.6. For example, in the sequence `04 35 EE`, the `04` indicates a loop instruction. The `35` indicates it is a `DBNE X` instruction (decrement register X and branch if result is not equal to zero), and the direction is backward (negative). The `EE` indicates a branch of `-18` bytes.
- Use up all the bytes for one instruction, then go on to the next instruction.

```
C6 05     => LDAA #$05      two-byte LDAA, IMM addressing mode
CE 20 00  => LDX  #$2000    three-byte LDX, IMM addressing mode
E6 01     => LDAB 1,X       two to four-byte LDAB, IDX addressing
                            mode.  Operand 01 => 1,X, a 5b constant
                            offset which uses only one postbyte
18 06     => ABA            two-byte ABA, INH addressing mode
04 35 EE  => DBNE X,(-18)   three-byte loop instruction
                            Postbyte 35 indicates DBNE X, negative
3F        => SWI            one-byte SWI, INH addressing mode
```

## Table A-2. CPU12 Opcode Map (Sheet 1 of 2)

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

**Key to Table A-2**



Opcode

Mnemonic

Address Mode

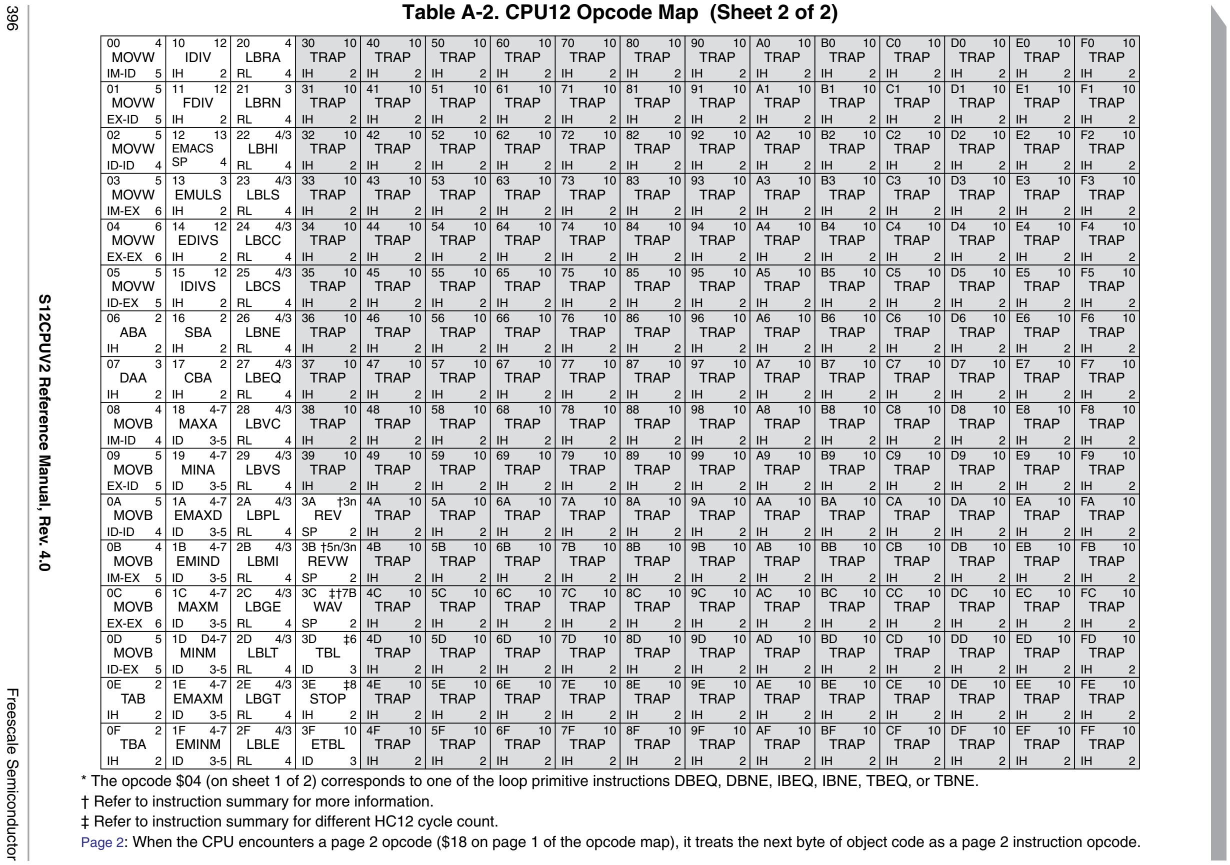
00 5

BGND

IH I

Number of HCS12 cycles (‡ indicates HC12 different)

Number of bytes



396

S12CPUV2 Reference Manual, Rev. 4.0

Freescale Semiconductor

### Table A-2. CPU12 Opcode Map (Sheet 2 of 2)

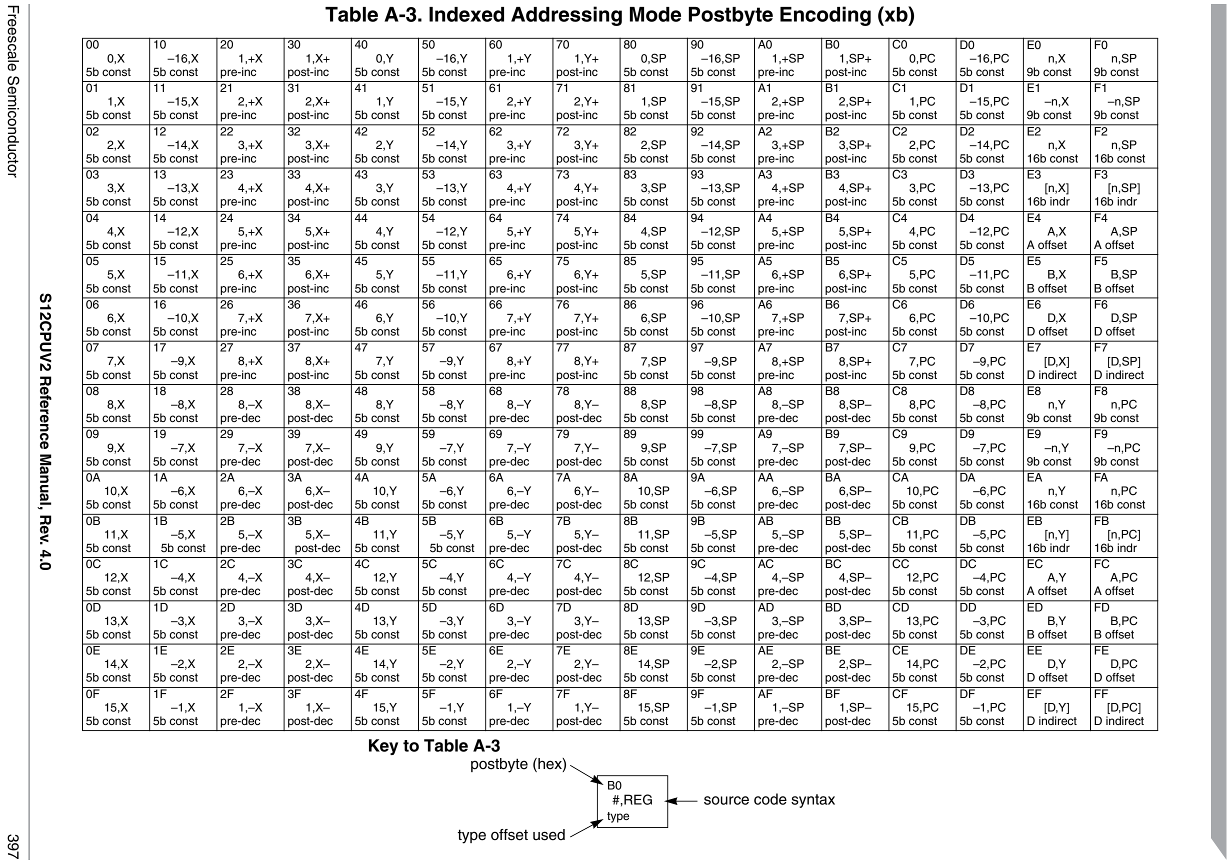
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 4<br>MOVW<br>IM-ID 5 | 10 12<br>IDIV<br>IH 2 | 20 4<br>LBRA<br>RL 4 | 30 10<br>TRAP<br>IH 2 | 40 10<br>TRAP<br>IH 2 | 50 10<br>TRAP<br>IH 2 | 60 10<br>TRAP<br>IH 2 | 70 10<br>TRAP<br>IH 2 | 80 10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

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

† Refer to instruction summary for more information.

‡ Refer to instruction summary for different HC12 cycle count.

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

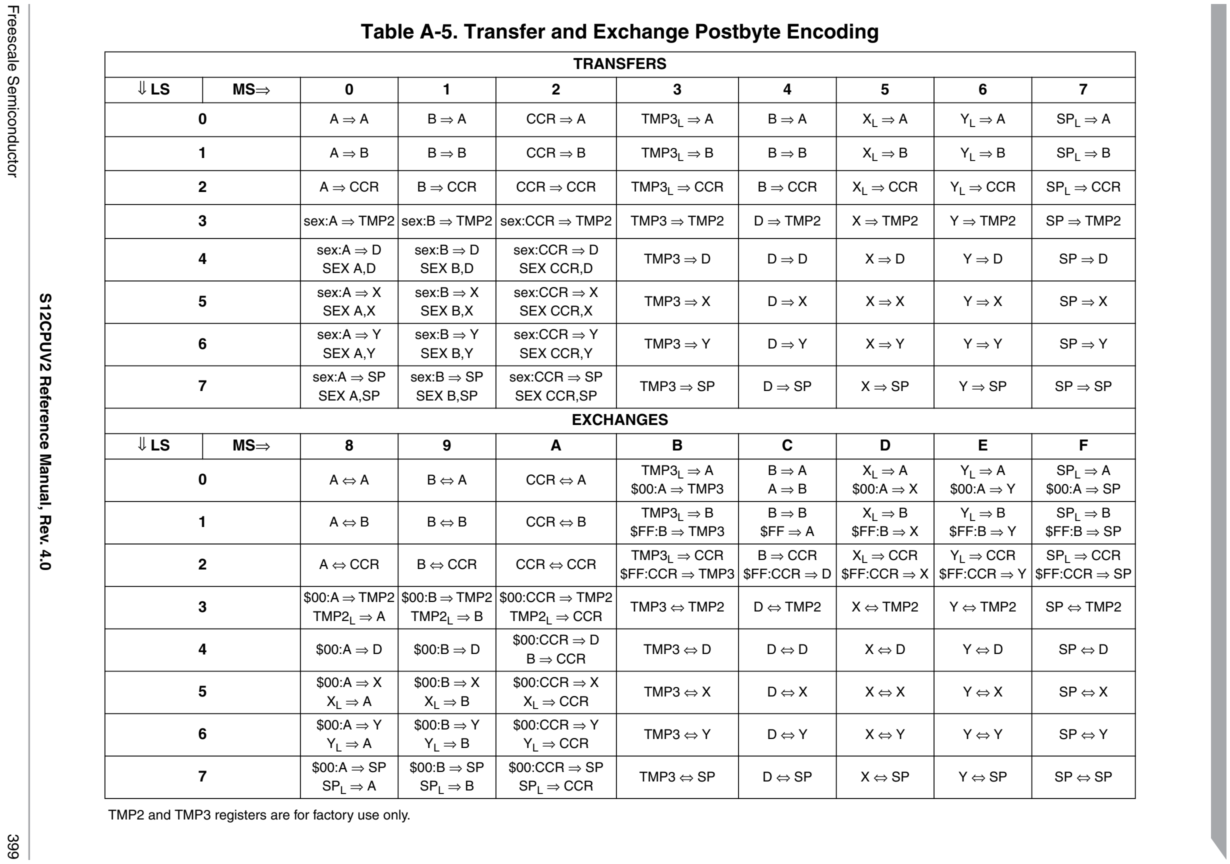
**Key to Table A-3**

postbyte (hex)

B0
#,REG
type

source code syntax

type offset used



## Table A-5. Transfer and Exchange Postbyte Encoding

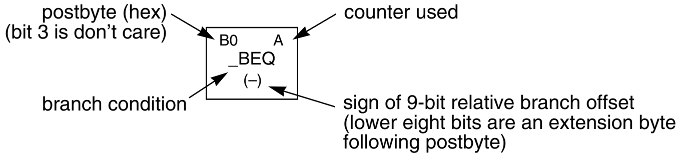
| TRANSFERS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ⇓LS | MS⇒ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | | A ⇒ A | B ⇒ A | CCR ⇒ A | $TMP3_L$ ⇒ A | B ⇒ A | $X_L$ ⇒ A | $Y_L$ ⇒ A | $SP_L$ ⇒ A |
| 1 | | A ⇒ B | B ⇒ B | CCR ⇒ B | $TMP3_L$ ⇒ B | B ⇒ B | $X_L$ ⇒ B | $Y_L$ ⇒ B | $SP_L$ ⇒ B |
| 2 | | A ⇒ CCR | B ⇒ CCR | CCR ⇒ CCR | $TMP3_L$ ⇒ CCR | B ⇒ CCR | $X_L$ ⇒ CCR | $Y_L$ ⇒ CCR | $SP_L$ ⇒ CCR |
| 3 | | sex:A ⇒ TMP2 | sex:B ⇒ TMP2 | sex:CCR ⇒ TMP2 | TMP3 ⇒ TMP2 | D ⇒ TMP2 | X ⇒ TMP2 | Y ⇒ TMP2 | SP ⇒ TMP2 |
| 4 | | sex:A ⇒ D<br>SEX A,D | sex:B ⇒ D<br>SEX B,D | sex:CCR ⇒ D<br>SEX CCR,D | TMP3 ⇒ D | D ⇒ D | X ⇒ D | Y ⇒ D | SP ⇒ D |
| 5 | | sex:A ⇒ X<br>SEX A,X | sex:B ⇒ X<br>SEX B,X | sex:CCR ⇒ X<br>SEX CCR,X | TMP3 ⇒ X | D ⇒ X | X ⇒ X | Y ⇒ X | SP ⇒ X |
| 6 | | sex:A ⇒ Y<br>SEX A,Y | sex:B ⇒ Y<br>SEX B,Y | sex:CCR ⇒ Y<br>SEX CCR,Y | TMP3 ⇒ Y | D ⇒ Y | X ⇒ Y | Y ⇒ Y | SP ⇒ Y |
| 7 | | sex:A ⇒ SP<br>SEX A,SP | sex:B ⇒ SP<br>SEX B,SP | sex:CCR ⇒ SP<br>SEX CCR,SP | TMP3 ⇒ SP | D ⇒ SP | X ⇒ SP | Y ⇒ SP | SP ⇒ SP |
| EXCHANGES | | | | | | | | | |
| ⇓LS | MS⇒ | 8 | 9 | A | B | C | D | E | F |
| 0 | | A ⇔ A | B ⇔ A | CCR ⇔ A | $TMP3_L$ ⇒ A<br>\$00:A ⇒ TMP3 | B ⇒ A<br>A ⇒ B | $X_L$ ⇒ A<br>\$00:A ⇒ X | $Y_L$ ⇒ A<br>\$00:A ⇒ Y | $SP_L$ ⇒ A<br>\$00:A ⇒ SP |
| 1 | | A ⇔ B | B ⇔ B | CCR ⇔ B | $TMP3_L$ ⇒ B<br>\$FF:B ⇒ TMP3 | B ⇒ B<br>\$FF ⇒ A | $X_L$ ⇒ B<br>\$FF:B ⇒ X | $Y_L$ ⇒ B<br>\$FF:B ⇒ Y | $SP_L$ ⇒ B<br>\$FF:B ⇒ SP |
| 2 | | A ⇔ CCR | B ⇔ CCR | CCR ⇔ CCR | $TMP3_L$ ⇒ CCR<br>\$FF:CCR ⇒ TMP3 | B ⇒ CCR<br>\$FF:CCR ⇒ D | $X_L$ ⇒ CCR<br>\$FF:CCR ⇒ X | $Y_L$ ⇒ CCR<br>\$FF:CCR ⇒ Y | $SP_L$ ⇒ CCR<br>\$FF:CCR ⇒ SP |
| 3 | | \$00:A ⇒ TMP2<br>$TMP2_L$ ⇒ A | \$00:B ⇒ TMP2<br>$TMP2_L$ ⇒ B | \$00:CCR ⇒ TMP2<br>$TMP2_L$ ⇒ CCR | TMP3 ⇔ TMP2 | D ⇔ TMP2 | X ⇔ TMP2 | Y ⇔ TMP2 | SP ⇔ TMP2 |
| 4 | | \$00:A ⇒ D | \$00:B ⇒ D | \$00:CCR ⇒ D<br>B ⇒ CCR | TMP3 ⇔ D | D ⇔ D | X ⇔ D | Y ⇔ D | SP ⇔ D |
| 5 | | \$00:A ⇒ X<br>$X_L$ ⇒ A | \$00:B ⇒ X<br>$X_L$ ⇒ B | \$00:CCR ⇒ X<br>$X_L$ ⇒ CCR | TMP3 ⇔ X | D ⇔ X | X ⇔ X | Y ⇔ X | SP ⇔ X |
| 6 | | \$00:A ⇒ Y<br>$Y_L$ ⇒ A | \$00:B ⇒ Y<br>$Y_L$ ⇒ B | \$00:CCR ⇒ Y<br>$Y_L$ ⇒ CCR | TMP3 ⇔ Y | D ⇔ Y | X ⇔ Y | Y ⇔ Y | SP ⇔ Y |
| 7 | | \$00:A ⇒ SP<br>$SP_L$ ⇒ A | \$00:B ⇒ SP<br>$SP_L$ ⇒ B | \$00:CCR ⇒ SP<br>$SP_L$ ⇒ CCR | TMP3 ⇔ SP | D ⇔ SP | X ⇔ SP | Y ⇔ SP | SP ⇔ SP |

TMP2 and TMP3 registers are for factory use only.

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

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

### 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.