UNIVERSITY OF CAPE TOWN

DEPARTMENT OF ELECTRICAL ENGINEERING

EEE226S/EEE233S/EEE234S/EEE370S

Module D: Measurement and Microprocessors

FINAL EXAMINATION NOVEMBER 2006

TIME: 2 hours

TOTAL MARKS: 100 (See Note 3 below)

INSTRUCTIONS

1. This is a Closed Book Examination. Candidates will be supplied with the Instruction Set for the 68HCS908GT16 and an ASCII table (attached).

2. All numerical answers must be given to the appropriate number of significant figures, and **the base of the number system must be indicated if it is not base 10**.

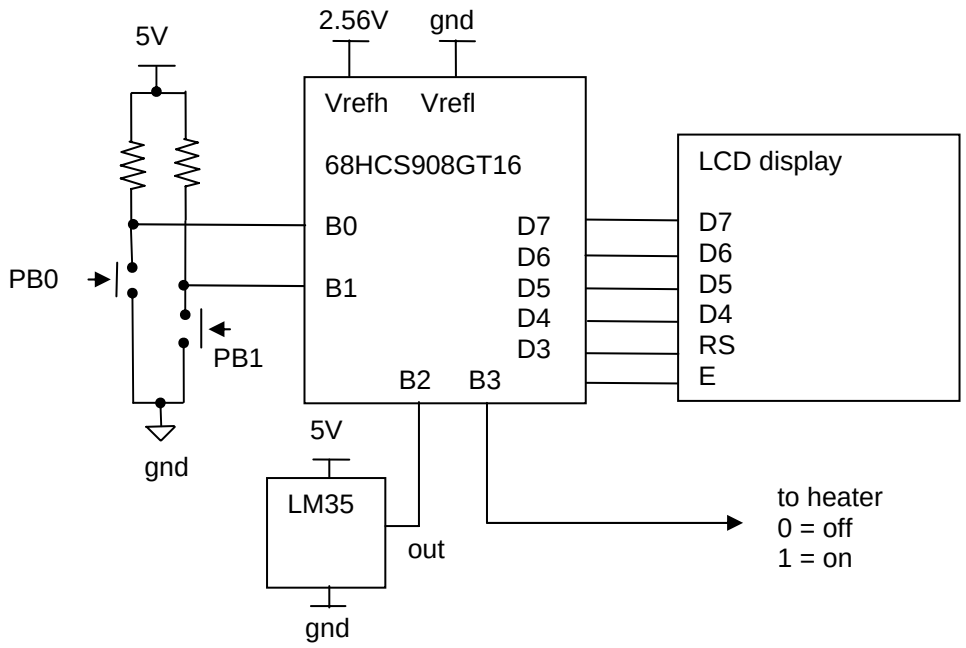
3. Answer all questions. There are 100 marks available. There is no sub-minimum in this course.

INTERNAL EXAMINER: PROF. J. TAPSON
EXTERNAL EXAMINER: DR. D. HOCH

Question 1:

A Freescale 68HCS908GT16 microprocessor is to be used to control the heating in a room. It will have a display that shows the desired temperature, and the actual temperature. To raise the desired temperature, one button is pushed. To lower the desired temperature, the other button is pushed.

The circuit is shown below. The processors is connected to push buttons on pins B0 and B1, and to an LCD display on pins D7, D6, D5, and D4, as shown below (all other connections such as power, ground, the oscillator and so on can be assumed to be standard). All resistors are 1kΩ.



The processor is connected to an LM35 analog temperature sensor via pin B2. The LM35 gives a voltage of 0.01V/°C at its output. Pin B3 is connected to a heater, where logic "0" turns the heater off, and logic "1" turns it on.

Write a program that works as follows:

- The temperature is displayed on the LCD display.
- A desired (setpoint) temperature is also displayed.
- Pushing button PB0 lowers the setpoint temperature by 1°C.
- Pushing button PB1 raises the setpoint temperature by 1°C.
- The processor turns the heater on and off depending on the measured and setpoint temperatures.

You may assume that you already have the following subroutines. You <u>do not</u> have to write them:

Delay_1sec:	this subroutine gives a one second delay.
Init_LCD:	Initialises the LCD.
Clear_LCD:	Clears the LCD.
Write_LCD1:	Writes the character in the accumulator to line 1 of the LCD.
Write_LCD2:	Writes the character in the accumulator to line 2 of the LCD.
GetADC_B2:	this subroutine gets an 8-bit ADC conversion from pin B2 and returns it in the accumulator.

a) Complete the following program to carry out the actions listed above. You may use all the defined variables and can define extra ones if you wish. You do not need to reproduce all the lines below in you answer book – just write down the missing code.

You do not need to define any of the variables used in the subroutines listed above.

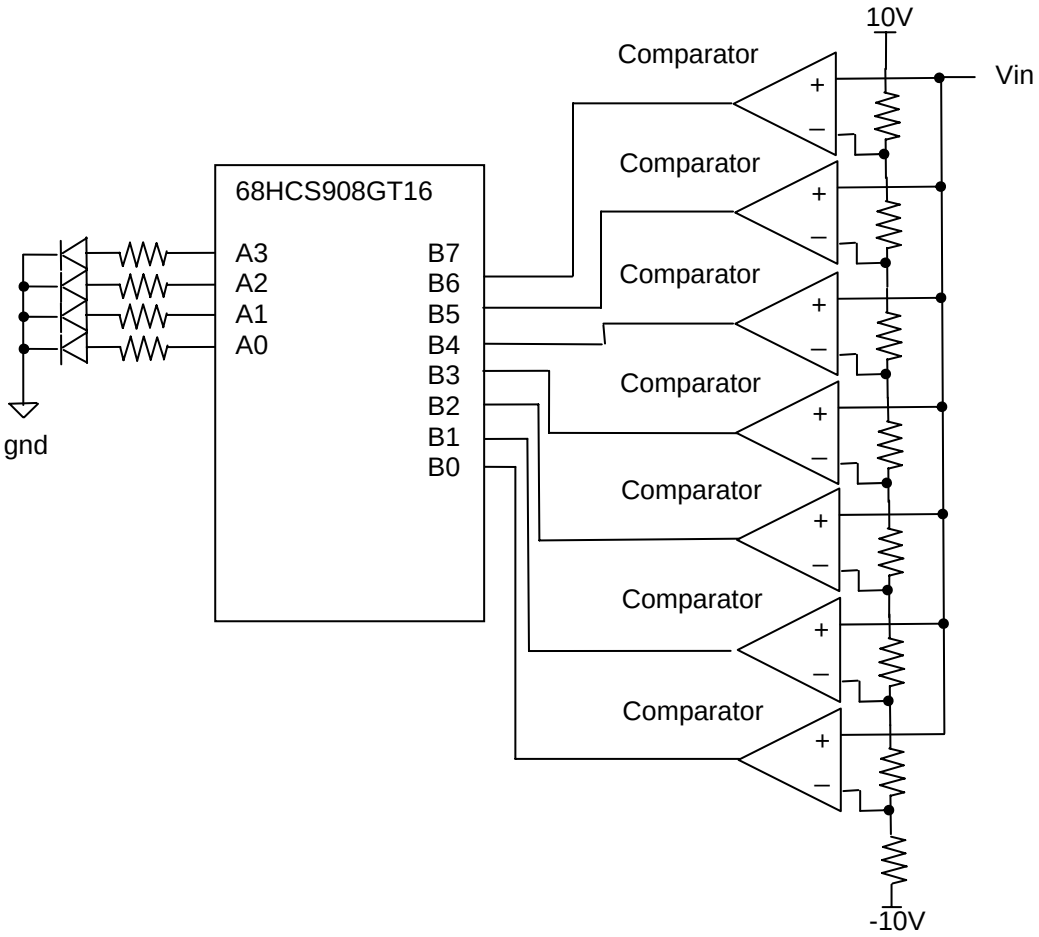
You must provide switch debouncing in your program.

```
RomStart        EQU     $C000           ; Start of program memory
RamStart        EQU     $0080           ; Start of data memory
PTB             EQU     $0004           ; Port B
PTBPE           EQU     $0006           ; Port B pullup enable
DDRB            EQU     $0007           ; Port B direction register
PTD             EQU     $000C           ; Port D
PTDPE           EQU     $000B           ; Port D pullup enable
DDRD            EQU     $000F           ; Port D direction register

                org     RamStart        ; Start variables here

Timeout1        ds    1                 ; Define a byte variable
Count           ds    1                 ; Define a byte variable

                org     RomStart        ; Start program here
main_init:

;****************** YOUR CODE STARTS HERE  **********************


;****************** END OF YOUR CODE  ****************************
```

total: (30)

Question 2:

A 68HCS908GT16 microprocessor is connected to a set of comparators and LEDs as shown in the diagram below. The comparators and resistors are set up to implement a flash ADC. All resistors are 1kΩ.



a) How many bits of resolution does the ADC have? (2)

b) What is the size of the least bit as a voltage? (3)

c) Why is there no EOC or clock line for this converter? (2)

d) Draw a truth table for the inputs B0-B7 and the ADC output. (6)

e) What is the output if Vin = 6.3 V? (2)

f) If the ADC output is to be displayed on the LEDs connected to Port A, as shown in the diagram, complete the following program:

```
RomStart        EQU     $C000           ; Start of program memory
RamStart        EQU     $0080           ; Start of data memory
PTA             EQU     $0000           ; Port A
PTAPE           EQU     $0002           ; Port A pullup enable
```

```
DDRA            EQU     $0003           ; Port A direction register
PTB             EQU     $0004           ; Port B
PTBPE           EQU     $0006           ; Port B pullup enable
DDRB            EQU     $0007           ; Port B direction register
PTD             EQU     $000C           ; Port D
PTDPE           EQU     $000B           ; Port D pullup enable
DDRD            EQU     $000F           ; Port D direction register
                org     RamStart        ; Start variables here

Timeout1        ds    1                 ; Define a byte variable
Count           ds    1                 ; Define a byte variable
Loop            ds    1                 ; Define a byte variable

                    org     RomStart        ; Start program here
main_init:

;****************** YOUR CODE STARTS HERE  **********************



;****************** END OF YOUR CODE  ****************************
```

(15)

g) Assuming the comparators are infinitely fast and take no time at all to do a comparison, what is the sampling speed of your converter, in instruction cycles?

(10)

total: (40)

Question 3:

The following code shows a subroutine designed to give varying delays.

```
      org   $0080 ; start of RAM
del1  ds    1
del2  ds    1

      org   $c000 ; Start of program memoroy

;*  some program here – ignore

;A set of delay routines of various lengths.
;call delay4 for a short delay, delay1 for an intermediate delay
;and delay5 for a long delay.

delay4:
       mov #$03,del1
       bra delay2
delay5:
       mov #$FF,del1
       bra delay2 ←————————
delay1:
        mov #$A0,del1
```

```
delay2: mov #$03,del2
delay3: nop
        nop
        nop
        dbnz del2,delay3
        dbnz del1,delay2
        rts
```

a) If the instructions take the following number of clock cycles:

Instruction	Cycles
rol	3
dbnz	5
mov	4
nop	1
jsr	6
bra	4
rts	4

how many cycles would it take from a program call **`jsr delay4`** to the end of the subroutine (**`rts`**), inclusive of the **`jsr`** and **`rts`** instructions? (10)

b) Write a subroutine which would give a delay of exactly four hundred cycles from **`jsr`** to **`rts`**. Show mathematically how your subroutine gives this length of delay. (12)

c) What would you expect the hexadecimal opcode and operands for the instruction **`mov #$A0,del1`** to be? (3)

d) What would you expect the hexadecimal opcode and operands for the line **`bra delay2`** (marked with an arrow in the program above) to be?

(5)

total: (30)

ASCII Table

Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex
(nul)	0	0x00	space	32	0x20	@	64	0x40	`	96	0x60
(soh)	1	0x01	!	33	0x21	A	65	0x41	a	97	0x61
(stx)	2	0x02	“	34	0x22	B	66	0x42		98	0x62
(etx)	3	0x03	#	35	0x23	C	67	0x43	c	99	0x63
(eot)	4	0x04	$	36	0x24	D	68	0x44	d	100	0x64
(enq)	5	0x05	%	37	0x25	E	69	0x45	e	101	0x65
(ack)	6	0x06	&	38	0x26	F	70	0x46	f	102	0x66
(bel)	7	0x07	‘	39	0x27	G	71	0x47	g	103	0x67
(bs)	8	0x08	(	40	0x28	H	72	0x48	h	104	0x68
(ht)	9	0x09	)	41	0x29	I	73	0x49	i	105	0x69
(nl)	10	0x0a	*	42	0x2a	J	74	0x4a	j	106	0x6a
(vt)	11	0x0b	+	43	0x2b	K	75	0x4b	k	107	0x6b
(np)	12	0x0c	,	44	0x2c	L	76	0x4c	l	108	0x6c
(cr)	13	0x0d	-	45	0x2d	M	77	0x4d	m	109	0x6d
(so)	14	0x0e	.	46	0x2e	N	78	0x4e	n	110	0x6e
(si)	15	0x0f	/	47	0x2f	O	79	0x4f	o	111	0x6f
(dle)	16	0x10	0	48	0x30	P	80	0x50	p	112	0x70
(dc1)	17	0x11	1	49	0x31	Q	81	0x51	q	113	0x71
(dc2)	18	0x12	2	50	0x32	R	82	0x52	r	114	0x72
(dc3)	19	0x13	3	51	0x33	S	83	0x53	s	115	0x73
(dc4)	20	0x14	4	52	0x34	T	84	0x54	t	116	0x74
(nak)	21	0x15	5	53	0x35	U	85	0x55	u	117	0x75
(syn)	22	0x16	6	54	0x36	V	86	0x56	v	118	0x76
(etb)	23	0x17	7	55	0x37	W	87	0x57	w	119	0x77
(can)	24	0x18	8	56	0x38	X	88	0x58	x	120	0x78
(em)	25	0x19	9	57	0x39	Y	89	0x59	y	121	0x79
(sub)	26	0x1a	:	58	0x3a	Z	90	0x5a	z	122	0x7a
(esc)	27	0x1b	;	59	0x3b	[	91	0x5b	{	123	0x7b
(fs)	28	0x1c	<	60	0x3c	\	92	0x5c	|	124	0x7c
(gs)	29	0x1d	=	61	0x3d	]	93	0x5d	}	125	0x7d
(rs)	30	0x1e	>	62	0x3e	^	94	0x5e	~	126	0x7e
(us)	31	0x1f	?	63	0x3f	_	95	0x5f	(del)	127	0x7f

Table 7-1 HCS08 Instruction Set Summary (Sheet 1 of 6)

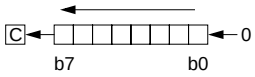
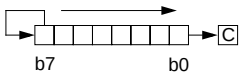
Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Bus Cycles[1]
ADC #*opr8i* ADC *opr8a* ADC *opr16a* ADC *oprx16*,X ADC *oprx8*,X ADC ,X ADC *oprx16*,SP ADC *oprx8*,SP	Add with Carry	A ← (A) + (M) + (C)			–				IMM DIR EXT IX2 IX1 IX SP2 SP1	A9 B9 C9 D9 E9 F9 9ED9 9EE9	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
ADD #*opr8i* ADD *opr8a* ADD *opr16a* ADD *oprx16*,X ADD *oprx8*,X ADD ,X ADD *oprx16*,SP ADD *oprx8*,SP	Add without Carry	A ← (A) + (M)			–				IMM DIR EXT IX2 IX1 IX SP2 SP1	AB BB CB DB EB FB 9EDB 9EEB	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
AIS #*opr8i*	Add Immediate Value (Signed) to Stack Pointer	SP ← (SP) + (M) M is sign extended to a 16-bit value	–	–	–	–	–	–	IMM	A7	ii	2
AIX #*opr8i*	Add Immediate Value (Signed) to Index Register (H:X)	H:X ← (H:X) + (M) M is sign extended to a 16-bit value	–	–	–	–	–	–	IMM	AF	ii	2
AND #*opr8i* AND *opr8a* AND *opr16a* AND *oprx16*,X AND *oprx8*,X AND ,X AND *oprx16*,SP AND *oprx8*,SP	Logical AND	A ← (A) & (M)	0	–	–			–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A4 B4 C4 D4 E4 F4 9ED4 9EE4	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
ASL *opr8a* ASLA ASLX ASL *oprx8*,X ASL ,X ASL *oprx8*,SP	Arithmetic Shift Left (Same as LSL)				–	–			DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	5 1 1 5 4 6
ASR *opr8a* ASRA ASRX ASR *oprx8*,X ASR ,X ASR *oprx8*,SP	Arithmetic Shift Right				–	–			DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	5 1 1 5 4 6
BCC *rel*	Branch if Carry Bit Clear	Branch if (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BCLR *n*,*opr8a*	Clear Bit n in Memory	Mn ← 0	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ *rel*	Branch if Equal	Branch if (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE *rel*	Branch if Greater Than or Equal To (Signed Operands)	Branch if $(N \oplus V) = 0$	–	–	–	–	–	–	REL	90	rr	3
BGND	Enter Active Background if ENBDM = 1	Waits For and Processes BDM Commands Until GO, TRACE1, or TAGGO	–	–	–	–	–	–	INH	82		5+
BGT *rel*	Branch if Greater Than (Signed Operands)	Branch if (Z) | $(N \oplus V) = 0$	–	–	–	–	–	–	REL	92	rr	3
BHCC *rel*	Branch if Half Carry Bit Clear	Branch if (H) = 0	–	–	–	–	–	–	REL	28	rr	3

Table 7-1 HCS08 Instruction Set Summary (Sheet 2 of 6)

Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Bus Cycles[1]
BHCS *rel*	Branch if Half Carry Bit Set	Branch if (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI *rel*	Branch if Higher	Branch if (C) | (Z) = 0	–	–	–	–	–	–	REL	22	rr	3
BHS *rel*	Branch if Higher or Same (Same as BCC)	Branch if (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if IRQ Pin High	Branch if IRQ pin = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if IRQ Pin Low	Branch if IRQ pin = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr8i* BIT *opr8a* BIT *opr16a* BIT *oprx16*,X BIT *oprx8*,X BIT ,X BIT *oprx16*,SP BIT *oprx8*,SP	Bit Test	(A) & (M) (CCR Updated but Operands Not Changed)	0	–	–			–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 B5 C5 D5 E5 F5 9ED5 9EE5	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
BLE *rel*	Branch if Less Than or Equal To (Signed Operands)	Branch if (Z) | (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO *rel*	Branch if Lower (Same as BCS)	Branch if (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	Branch if (C) | (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *rel*	Branch if Less Than (Signed Operands)	Branch if (N ⊕ V) = 1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	Branch if (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	Branch if (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	Branch if (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	Branch if (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	Branch if (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	No Test	–	–	–	–	–	–	REL	20	rr	3
BRCLR *n*,*opr8a*,*rel*	Branch if Bit *n* in Memory Clear	Branch if (Mn) = 0	–	–	–	–	–		DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	Uses 3 Bus Cycles	–	–	–	–	–	–	REL	21	rr	3
BRSET *n*,*opr8a*,*rel*	Branch if Bit *n* in Memory Set	Branch if (Mn) = 1	–	–	–	–	–		DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n*,*opr8a*	Set Bit *n* in Memory	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BSR *rel*	Branch to Subroutine	PC ← (PC) + $0002 push (PCL); SP ← (SP) – $0001 push (PCH); SP ← (SP) – $0001 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	5

Table 7-1 HCS08 Instruction Set Summary (Sheet 3 of 6)

Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Bus Cycles[1]
CBEQ *opr8a*,*rel* CBEQA #*opr8i*,*rel* CBEQX #*opr8i*,*rel* CBEQ *oprx8*,X+,*rel* CBEQ ,X+,*rel* CBEQ *oprx8*,SP,*rel*	Compare and Branch if Equal	Branch if (A) = (M) Branch if (A) = (M) Branch if (X) = (M) Branch if (A) = (M) Branch if (A) = (M) Branch if (A) = (M)	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 5 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask Bit	I ← 0	–	–	0	–	–	–	INH	9A		1
CLR *opr8a* CLRA CLRX CLRH CLR *oprx8*,X CLR ,X CLR *oprx8*,SP	Clear	M ← $00 A ← $00 X ← $00 H ← $00 M ← $00 M ← $00 M ← $00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	5 1 1 1 5 4 6
CMP #*opr8i* CMP *opr8a* CMP *opr16a* CMP *oprx16*,X CMP *oprx8*,X CMP ,X CMP *oprx16*,SP CMP *oprx8*,SP	Compare Accumulator with Memory	(A) – (M) (CCR Updated But Operands Not Changed)		–	–				IMM DIR EXT IX2 IX1 IX SP2 SP1	A1 B1 C1 D1 E1 F1 9ED1 9EE1	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
COM *opr8a* COMA COMX COM *oprx8*,X COM ,X COM *oprx8*,SP	Complement (One's Complement)	M ← ($\overline{M}$)= $FF – (M) A ← ($\overline{A}$) = $FF – (A) X ← ($\overline{X}$) = $FF – (X) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M)	0	–	–			1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	5 1 1 5 4 6
CPHX *opr16a* CPHX #*opr16i* CPHX *opr8a* CPHX *oprx8*,SP	Compare Index Register (H:X) with Memory	(H:X) – (M:M + $0001) (CCR Updated But Operands Not Changed)		–	–				EXT IMM DIR SP1	3E 65 75 9EF3	hh ll jj kk dd ff	6 3 5 6
CPX #*opr8i* CPX *opr8a* CPX *opr16a* CPX *oprx16*,X CPX *oprx8*,X CPX ,X CPX *oprx16*,SP CPX *oprx8*,SP	Compare X (Index Register Low) with Memory	(X) – (M) (CCR Updated But Operands Not Changed)		–	–				IMM DIR EXT IX2 IX1 IX SP2 SP1	A3 B3 C3 D3 E3 F3 9ED3 9EE3	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	$(A)_{10}$	U	–	–				INH	72		1
DBNZ *opr8a*,*rel* DBNZA *rel* DBNZX *rel* DBNZ *oprx8*,X,*rel* DBNZ ,X,*rel* DBNZ *oprx8*,SP,*rel*	Decrement and Branch if Not Zero	Decrement A, X, or M Branch if (result) ≠ 0 DBNZX Affects X Not H	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	7 4 4 7 6 8
DEC *opr8a* DECA DECX DEC *oprx8*,X DEC ,X DEC *oprx8*,SP	Decrement	M ← (M) – $01 A ← (A) – $01 X ← (X) – $01 M ← (M) – $01 M ← (M) – $01 M ← (M) – $01		–	–			–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	5 1 1 5 4 6
DIV	Divide	A ← (H:A)÷(X) H ← Remainder	–	–	–	–			INH	52		6

Table 7-1 HCS08 Instruction Set Summary (Sheet 4 of 6)

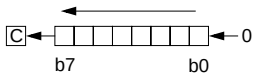
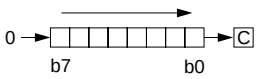
Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Bus Cycles[1]
EOR #*opr8i* EOR *opr8a* EOR *opr16a* EOR *oprx16*,X EOR *oprx8*,X EOR ,X EOR *oprx16*,SP EOR *oprx8*,SP	Exclusive OR Memory with Accumulator	A ← (A ⊕ M)	0	–	–			–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 B8 C8 D8 E8 F8 9ED8 9EE8	ii dd hh ll ee ff ff	2 3 4 4 3 3 5 4
INC *opr8a* INCA INCX INC *oprx8*,X INC ,X INC *oprx8*,SP	Increment	M ← (M) + $01 A ← (A) + $01 X ← (X) + $01 M ← (M) + $01 M ← (M) + $01 M ← (M) + $01		–	–			–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	5 1 1 5 4 6
JMP *opr8a* JMP *opr16a* JMP *oprx16*,X JMP *oprx8*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	3 4 4 3 3
JSR *opr8a* JSR *opr16a* JSR *oprx16*,X JSR *oprx8*,X JSR ,X	Jump to Subroutine	PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – $0001 Push (PCH); SP ← (SP) – $0001 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	5 6 6 5 5
LDA #*opr8i* LDA *opr8a* LDA *opr16a* LDA *oprx16*,X LDA *oprx8*,X LDA ,X LDA *oprx16*,SP LDA *oprx8*,SP	Load Accumulator from Memory	A ← (M)	0	–	–			–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A6 B6 C6 D6 E6 F6 9ED6 9EE6	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
LDHX #*opr16i* LDHX *opr8a* LDHX *opr16a* LDHX ,X LDHX *oprx16*,X LDHX *oprx8*,X LDHX *oprx8*,SP	Load Index Register (H:X) from Memory	H:X ← (M:M + $0001)	0	–	–			–	IMM DIR EXT IX IX2 IX1 SP1	45 55 32 9EAE 9EBE 9ECE 9EFE	jj kk dd hh ll ee ff ff ff	3 4 5 5 6 5 5
LDX #*opr8i* LDX *opr8a* LDX *opr16a* LDX *oprx16*,X LDX *oprx8*,X LDX ,X LDX *oprx16*,SP LDX *oprx8*,SP	Load X (Index Register Low) from Memory	X ← (M)	0	–	–			–	IMM DIR EXT IX2 IX1 IX SP2 SP1	AE BE CE DE EE FE 9EDE 9EEE	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
LSL *opr8a* LSLA LSLX LSL *oprx8*,X LSL ,X LSL *oprx8*,SP	Logical Shift Left (Same as ASL)			–	–				DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	5 1 1 5 4 6
LSR *opr8a* LSRA LSR*X* LSR *oprx8*,X LSR ,X LSR *oprx8*,SP	Logical Shift Right			–	–	0			DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	5 1 1 5 4 6
MOV *opr8a*,*opr8a* MOV *opr8a*,X+ MOV #*opr8i*,*opr8a* MOV ,X+,*opr8a*	Move	$(M)_{destination} \leftarrow (M)_{source}$ H:X ← (H:X) + $0001 in IX+/DIR and DIR/IX+ Modes	0	–	–			–	DIR/DIR DIR/IX+ IMM/DIR IX+/DIR	4E 5E 6E 7E	dd dd dd ii dd dd	5 5 4 5
MUL	Unsigned multiply	X:A ← (X) × (A)	–	0	–	–	–	0	INH	42		5

Table 7-1 HCS08 Instruction Set Summary (Sheet 5 of 6)

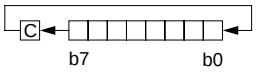
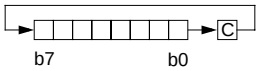
Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Bus Cycles[1]
NEG *opr8a* NEGA NEGX NEG *oprx8*,X NEG ,X NEG *oprx8*,SP	Negate (Two's Complement)	M ← – (M) = $00 – (M) A ← – (A) = $00 – (A) X ← – (X) = $00 – (X) M ← – (M) = $00 – (M) M ← – (M) = $00 – (M) M ← – (M) = $00 – (M)		–	–				DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	5 1 1 5 4 6
NOP	No Operation	Uses 1 Bus Cycle	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap Accumulator	A ← (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	62		1
ORA #*opr8i* ORA *opr8a* ORA *opr16a* ORA *oprx16*,X ORA *oprx8*,X ORA ,X ORA *oprx16*,SP ORA *oprx8*,SP	Inclusive OR Accumulator and Memory	A ← (A) | (M)	0	–	–			–	IMM DIR EXT IX2 IX1 IX SP2 SP1	AA BA CA DA EA FA 9EDA 9EEA	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
PSHA	Push Accumulator onto Stack	Push (A); SP ← (SP) – $0001	–	–	–	–	–	–	INH	87		2
PSHH	Push H (Index Register High) onto Stack	Push (H); SP ← (SP) – $0001	–	–	–	–	–	–	INH	8B		2
PSHX	Push X (Index Register Low) onto Stack	Push (X); SP ← (SP) – $0001	–	–	–	–	–	–	INH	89		2
PULA	Pull Accumulator from Stack	SP ← (SP + $0001); Pull (A)	–	–	–	–	–	–	INH	86		3
PULH	Pull H (Index Register High) from Stack	SP ← (SP + $0001); Pull (H)	–	–	–	–	–	–	INH	8A		3
PULX	Pull X (Index Register Low) from Stack	SP ← (SP + $0001); Pull (X)	–	–	–	–	–	–	INH	88		3
ROL *opr8a* ROLA ROLX ROL *oprx8*,X ROL ,X ROL *oprx8*,SP	Rotate Left through Carry			–	–				DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	5 1 1 5 4 6
ROR *opr8a* RORA RORX ROR *oprx8*,X ROR ,X ROR *oprx8*,SP	Rotate Right through Carry			–	–				DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	5 1 1 5 4 6
RSP	Reset Stack Pointer	SP ← $FF (High Byte Not Affected)	–	–	–	–	–	–	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + $0001; Pull (CCR) SP ← (SP) + $0001; Pull (A) SP ← (SP) + $0001; Pull (X) SP ← (SP) + $0001; Pull (PCH) SP ← (SP) + $0001; Pull (PCL)							INH	80		9
RTS	Return from Subroutine	SP ← SP + $0001; Pull (PCH) SP ← SP + $0001; Pull (PCL)	–	–	–	–	–	–	INH	81		6
SBC #*opr8i* SBC *opr8a* SBC *opr16a* SBC *oprx16*,X SBC *oprx8*,X SBC ,X SBC *oprx16*,SP SBC *oprx8*,SP	Subtract with Carry	A ← (A) – (M) – (C)		–	–				IMM DIR EXT IX2 IX1 IX SP2 SP1	A2 B2 C2 D2 E2 F2 9ED2 9EE2	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
SEC	Set Carry Bit	C ← 1	–	–	–	–	–	1	INH	99		1
SEI	Set Interrupt Mask Bit	I ← 1	–	–	1	–	–	–	INH	9B		1

Table 7-1 HCS08 Instruction Set Summary (Sheet 6 of 6)

Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Bus Cycles[1]
STA *opr8a* STA *opr16a* STA *oprx16*,X STA *oprx8*,X STA ,X STA *oprx16*,SP STA *oprx8*,SP	Store Accumulator in Memory	M ← (A)	0	–	–			–	DIR EXT IX2 IX1 IX SP2 SP1	B7 C7 D7 E7 F7 9ED7 9EE7	dd hh ll ee ff ff	3 4 4 3 2 5 4
STHX *opr8a* STHX *opr16a* STHX *oprx8*,SP	Store H:X (Index Reg.)	(M:M + $0001) ← (H:X)	0	–	–			–	DIR EXT SP1	35 96 9EFF	dd hh ll ff	4 5 5
STOP	Enable Interrupts: Stop Processing Refer to MCU Documentation	I bit ← 0; Stop Processing	–	–	0	–	–	–	INH	8E		2+
STX *opr8a* STX *opr16a* STX *oprx16*,X STX *oprx8*,X STX ,X STX *oprx16*,SP STX *oprx8*,SP	Store X (Low 8 Bits of Index Register) in Memory	M ← (X)	0	–	–			–	DIR EXT IX2 IX1 IX SP2 SP1	BF CF DF EF FF 9EDF 9EEF	dd hh ll ee ff ff ee ff ff	3 4 4 3 2 5 4
SUB #*opr8i* SUB *opr8a* SUB *opr16a* SUB *oprx16*,X SUB *oprx8*,X SUB ,X SUB *oprx16*,SP SUB *oprx8*,SP	Subtract	A ← (A) – (M)		–	–				IMM DIR EXT IX2 IX1 IX SP2 SP1	A0 B0 C0 D0 E0 F0 9ED0 9EE0	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
SWI	Software Interrupt	PC ← (PC) + $0001 Push (PCL); SP ← (SP) – $0001 Push (PCH); SP ← (SP) – $0001 Push (X); SP ← (SP) – $0001 Push (A); SP ← (SP) – $0001 Push (CCR); SP ← (SP) – $0001 I ← 1; PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	–	–	1	–	–	–	INH	83		11
TAP	Transfer Accumulator to CCR	CCR ← (A)							INH	84		1
TAX	Transfer Accumulator to X (Index Register Low)	X ← (A)	–	–	–	–	–	–	INH	97		1
TPA	Transfer CCR to Accumulator	A ← (CCR)	–	–	–	–	–	–	INH	85		1
TST *opr8a* TSTA TSTX TST *oprx8*,X TST ,X TST *oprx8*,SP	Test for Negative or Zero	(M) – $00 (A) – $00 (X) – $00 (M) – $00 (M) – $00 (M) – $00	0	–	–			–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	4 1 1 4 3 5
TSX	Transfer SP to Index Reg.	H:X ← (SP) + $0001	–	–	–	–	–	–	INH	95		2
TXA	Transfer X (Index Reg. Low) to Accumulator	A ← (X)	–	–	–	–	–	–	INH	9F		1
TXS	Transfer Index Reg. to SP	SP ← (H:X) – $0001	–	–	–	–	–	–	INH	94		2
WAIT	Enable Interrupts; Wait for Interrupt	I bit ← 0; Halt CPU	–	–	0	–	–	–	INH	8F		2+

NOTES:

1. Bus clock frequency is one-half of the CPU clock frequency.

Table 7-2 Opcode Map (Sheet 1 of 2)

Bit-Manipulation		Branch	Read-Modify-Write					Control		Register/Memory					
00 5 BRSET0 3 DIR	10 5 BSET0 2 DIR	20 3 BRA 2 REL	30 5 NEG 2 DIR	40 1 NEGA 1 INH	50 1 NEGX 1 INH	60 5 NEG 2 IX1	70 4 NEG 1 IX	80 9 RTI 1 INH	90 3 BGE 2 REL	A0 2 SUB 2 IMM	B0 3 SUB 2 DIR	C0 4 SUB 3 EXT	D0 4 SUB 3 IX2	E0 3 SUB 2 IX1	F0 3 SUB 1 IX
01 5 BRCLR0 3 DIR	11 5 BCLR0 2 DIR	21 3 BRN 2 REL	31 5 CBEQ 3 DIR	41 4 CBEQA 3 IMM	51 4 CBEQX 3 IMM	61 5 CBEQ 3 IX1+	71 5 CBEQ 2 IX+	81 6 RTS 1 INH	91 3 BLT 2 REL	A1 2 CMP 2 IMM	B1 3 CMP 2 DIR	C1 4 CMP 3 EXT	D1 4 CMP 3 IX2	E1 3 CMP 2 IX1	F1 3 CMP 1 IX
02 5 BRSET1 3 DIR	12 5 BSET1 2 DIR	22 3 BHI 2 REL	32 5 LDHX 3 EXT	42 5 MUL 1 INH	52 6 DIV 1 INH	62 1 NSA 1 INH	72 1 DAA 1 INH	82 5+ BGND 1 INH	92 3 BGT 2 REL	A2 2 SBC 2 IMM	B2 3 SBC 2 DIR	C2 4 SBC 3 EXT	D2 4 SBC 3 IX2	E2 3 SBC 2 IX1	F2 3 SBC 1 IX
03 5 BRCLR1 3 DIR	13 5 BCLR1 2 DIR	23 3 BLS 2 REL	33 5 COM 2 DIR	43 1 COMA 1 INH	53 1 COMX 1 INH	63 5 COM 2 IX1	73 4 COM 1 IX	83 11 SWI 1 INH	93 3 BLE 2 REL	A3 2 CPX 2 IMM	B3 3 CPX 2 DIR	C3 4 CPX 3 EXT	D3 4 CPX 3 IX2	E3 3 CPX 2 IX1	F3 3 CPX 1 IX
04 5 BRSET2 3 DIR	14 5 BSET2 2 DIR	24 3 BCC 2 REL	34 5 LSR 2 DIR	44 1 LSRA 1 INH	54 1 LSRX 1 INH	64 5 LSR 2 IX1	74 4 LSR 1 IX	84 1 TAP 1 INH	94 2 TXS 1 INH	A4 2 AND 2 IMM	B4 3 AND 2 DIR	C4 4 AND 3 EXT	D4 4 AND 3 IX2	E4 3 AND 2 IX1	F4 3 AND 1 IX
05 5 BRCLR2 3 DIR	15 5 BCLR2 2 DIR	25 3 BCS 2 REL	35 4 STHX 2 DIR	45 3 LDHX 3 IMM	55 4 LDHX 2 DIR	65 3 CPHX 3 IMM	75 5 CPHX 2 DIR	85 1 TPA 1 INH	95 2 TSX 1 INH	A5 2 BIT 2 IMM	B5 3 BIT 2 DIR	C5 4 BIT 3 EXT	D5 4 BIT 3 IX2	E5 3 BIT 2 IX1	F5 3 BIT 1 IX
06 5 BRSET3 3 DIR	16 5 BSET3 2 DIR	26 3 BNE 2 REL	36 5 ROR 2 DIR	46 1 RORA 1 INH	56 1 RORX 1 INH	66 5 ROR 2 IX1	76 4 ROR 1 IX	86 3 PULA 1 INH	96 5 STHX 3 EXT	A6 2 LDA 2 IMM	B6 3 LDA 2 DIR	C6 4 LDA 3 EXT	D6 4 LDA 3 IX2	E6 3 LDA 2 IX1	F6 3 LDA 1 IX
07 5 BRCLR3 3 DIR	17 5 BCLR3 2 DIR	27 3 BEQ 2 REL	37 5 ASR 2 DIR	47 1 ASRA 1 INH	57 1 ASRX 1 INH	67 5 ASR 2 IX1	77 4 ASR 1 IX	87 2 PSHA 1 INH	97 1 TAX 1 INH	A7 2 AIS 2 IMM	B7 3 STA 2 DIR	C7 4 STA 3 EXT	D7 4 STA 3 IX2	E7 3 STA 2 IX1	F7 2 STA 1 IX
08 5 BRSET4 3 DIR	18 5 BSET4 2 DIR	28 3 BHCC 2 REL	38 5 LSL 2 DIR	48 1 LSLA 1 INH	58 1 LSLX 1 INH	68 5 LSL 2 IX1	78 4 LSL 1 IX	88 3 PULX 1 INH	98 1 CLC 1 INH	A8 2 EOR 2 IMM	B8 3 EOR 2 DIR	C8 4 EOR 3 EXT	D8 4 EOR 3 IX2	E8 3 EOR 2 IX1	F8 3 EOR 1 IX
09 5 BRCLR4 3 DIR	19 5 BCLR4 2 DIR	29 3 BHCS 2 REL	39 5 ROL 2 DIR	49 1 ROLA 1 INH	59 1 ROLX 1 INH	69 5 ROL 2 IX1	79 4 ROL 1 IX	89 2 PSHX 1 INH	99 1 SEC 1 INH	A9 2 ADC 2 IMM	B9 3 ADC 2 DIR	C9 4 ADC 3 EXT	D9 4 ADC 3 IX2	E9 3 ADC 2 IX1	F9 3 ADC 1 IX
0A 5 BRSET5 3 DIR	1A 5 BSET5 2 DIR	2A 3 BPL 2 REL	3A 5 DEC 2 DIR	4A 1 DECA 1 INH	5A 1 DECX 1 INH	6A 5 DEC 2 IX1	7A 4 DEC 1 IX	8A 3 PULH 1 INH	9A 1 CLI 1 INH	AA 2 ORA 2 IMM	BA 3 ORA 2 DIR	CA 4 ORA 3 EXT	DA 4 ORA 3 IX2	EA 3 ORA 2 IX1	FA 3 ORA 1 IX
0B 5 BRCLR5 3 DIR	1B 5 BCLR5 2 DIR	2B 3 BMI 2 REL	3B 7 DBNZ 3 DIR	4B 4 DBNZA 2 INH	5B 4 DBNZX 2 INH	6B 7 DBNZ 3 IX1	7B 6 DBNZ 2 IX	8B 2 PSHH 1 INH	9B 1 SEI 1 INH	AB 2 ADD 2 IMM	BB 3 ADD 2 DIR	CB 4 ADD 3 EXT	DB 4 ADD 3 IX2	EB 3 ADD 2 IX1	FB 3 ADD 1 IX
0C 5 BRSET6 3 DIR	1C 5 BSET6 2 DIR	2C 3 BMC 2 REL	3C 5 INC 2 DIR	4C 1 INCA 1 INH	5C 1 INCX 1 INH	6C 5 INC 2 IX1	7C 4 INC 1 IX	8C 1 CLRH 1 INH	9C 1 RSP 1 INH		BC 3 JMP 2 DIR	CC 4 JMP 3 EXT	DC 4 JMP 3 IX2	EC 3 JMP 2 IX1	FC 3 JMP 1 IX
0D 5 BRCLR6 3 DIR	1D 5 BCLR6 2 DIR	2D 3 BMS 2 REL	3D 4 TST 2 DIR	4D 1 TSTA 1 INH	5D 1 TSTX 1 INH	6D 4 TST 2 IX1	7D 3 TST 1 IX		9D 1 NOP 1 INH	AD 5 BSR 2 REL	BD 5 JSR 2 DIR	CD 6 JSR 3 EXT	DD 6 JSR 3 IX2	ED 5 JSR 2 IX1	FD 5 JSR 1 IX
0E 5 BRSET7 3 DIR	1E 5 BSET7 2 DIR	2E 3 BIL 2 REL	3E 6 CPHX 3 EXT	4E 6 MOV 3 DD	5E 5 MOV 2 DIX+	6E 4 MOV 3 IMD	7E 5 MOV 2 IX+D	8E 2+ STOP 1 INH	9E Page 2	AE 2 LDX 2 IMM	BE 3 LDX 2 DIR	CE 4 LDX 3 EXT	DE 4 LDX 3 IX2	EE 3 LDX 2 IX1	FE 3 LDX 1 IX
0F 5 BRCLR7 3 DIR	1F 5 BCLR7 2 DIR	2F 3 BIH 2 REL	3F 5 CLR 2 DIR	4F 1 CLRA 1 INH	5F 1 CLRX 1 INH	6F 5 CLR 2 IX1	7F 4 CLR 1 IX	8F 2+ WAIT 1 INH	9F 1 TXA 1 INH	AF 2 AIX 2 IMM	BF 3 STX 2 DIR	CF 4 STX 3 EXT	DF 4 STX 3 IX2	EF 3 STX 2 IX1	FF 2 STX 1 IX

INH	**Inherent**	**REL**	**Relative**	**SP1**	**Stack Pointer, 8-Bit Offset**
IMM	**Immediate**	**IX**	**Indexed, No Offset**	**SP2**	**Stack Pointer, 16-Bit Offset**
DIR	**Direct**	**IX1**	**Indexed, 8-Bit Offset**	**IX+**	**Indexed, No Offset with Post Increment**
EXT	**Extended**	**IX2**	**Indexed, 16-Bit Offset**		
DD	**DIR to DIR**	**IMD**	**IMM to DIR**	**IX1+**	**Indexed, 1-Byte Offset with Post Increment**
IX+D	**IX+ to DIR**	**DIX+**	**DIR to IX+**		

Opcode in Hexadecimal	F0 3	**HCS08 Cycles**
	SUB	**Instruction Mnemonic**
Number of Bytes	1 IX	**Addressing Mode**

Table 7-2 Opcode Map (Sheet 2 of 2)

Bit-Manipulation		Branch	Read-Modify-Write					Control		Register/Memory					
						9E60 6 NEG 3 SP1							9ED0 5 SUB 4 SP2	9EE0 4 SUB 3 SP1	
						9E61 6 CBEQ 4 SP1							9ED1 5 CMP 4 SP2	9EE1 4 CMP 3 SP1	
													9ED2 5 SBC 4 SP2	9EE2 4 SBC 3 SP1	
						9E63 6 COM 3 SP1							9ED3 5 CPX 4 SP2	9EE3 4 CPX 3 SP1	9EF3 6 CPHX 3 SP1
						9E64 6 LSR 3 SP1							9ED4 5 AND 4 SP2	9EE4 4 AND 3 SP1	
													9ED5 5 BIT 4 SP2	9EE5 4 BIT 3 SP1	
						9E66 6 ROR 3 SP1							9ED6 5 LDA 4 SP2	9EE6 4 LDA 3 SP1	
						9E67 6 ASR 3 SP1							9ED7 5 STA 4 SP2	9EE7 4 STA 3 SP1	
						9E68 6 LSL 3 SP1							9ED8 5 EOR 4 SP2	9EE8 4 EOR 3 SP1	
						9E69 6 ROL 3 SP1							9ED9 5 ADC 4 SP2	9EE9 4 ADC 3 SP1	
						9E6A 6 DEC 3 SP1							9EDA 5 ORA 4 SP2	9EEA 4 ORA 3 SP1	
						9E6B 8 DBNZ 4 SP1							9EDB 5 ADD 4 SP2	9EEB 4 ADD 3 SP1	
						9E6C 6 INC 3 SP1									
						9E6D 5 TST 3 SP1									
										9EAE 5 LDHX 2 IX	9EBE 6 LDHX 4 IX2	9ECE 5 LDHX 3 IX1	9EDE 5 LDX 4 SP2	9EEE 4 LDX 3 SP1	9EFE 5 LDHX 3 SP1
						9E6F 6 CLR 3 SP1							9EDF 5 STX 4 SP2	9EEF 4 STX 3 SP1	9EFF 5 STHX 3 SP1

INH	**Inherent**	**REL**	**Relative**	**SP1**	**Stack Pointer, 8-Bit Offset**
IMM	**Immediate**	**IX**	**Indexed, No Offset**	**SP2**	**Stack Pointer, 16-Bit Offset**
DIR	**Direct**	**IX1**	**Indexed, 8-Bit Offset**	**IX+**	**Indexed, No Offset with Post Increment**
EXT	**Extended**	**IX2**	**Indexed, 16-Bit Offset**		
DD	**DIR to DIR**	**IMD**	**IMM to DIR**	**IX1+**	**Indexed, 1-Byte Offset with Post Increment**
IX+D	**IX+ to DIR**	**DIX+**	**DIR to IX+**		

Note: All Sheet 2 Opcodes are Preceded by the Page 2 Prebyte (9E)

Prebyte (9E) and Opcode in Hexadecimal
Number of Bytes

9E60	6
NEG	
3	SP1

HCS08 Cycles
Instruction Mnemonic
Addressing Mode