6809/6309 Assembly and Mnemonic Information

Compiled and edited by Chris Lomont, www.lomont.org. Version 1.1 April 2007

* denotes 6309 only instruction, ~/~ is cycle counts on 6809/6309, # is bytes,
~ and # can be increased by addressing and other factors, see throughout
! prefix opcode with 10, @ prefix opcode with 11, e.g., !8B is opcode 10 8B
CCodes condition codes (6809 only for now): * affected, - not, ? indeterminate
I is interrupt flag: E = bit 7, F=FIRQ bit6, I IRQ bit 4, notes later
Indexed and Extended cycle counts and byte length modified by mode

Mnem	Immed.			Direct			Indexed			Extended			Inherent			CCodes 53210
	OP	~/~	#	OP	~/~	+	OP	~/~	#	OP	~/~	#	OP	~/~	#	IHNZVC
ABX													3A	3/1	1	------
ADCA	89	2	2	99	4/3	2	A9	4+	2+	B9	5/4	3				-*****
ADCB	C9	2	2	D9	4/3	2	E9	4+	2+	F9	5/3	3				-*****
*ADCD	!89	5/4	4	!99	7/5	3	!A9	7+/6+	3+	!B9	8/6	4				
ADDA	8B	2	2	9B	4/3	2	AB	4+	2+	BB	5/4	3				-*****
ADDB	CB	2	2	DB	4/3	2	EB	4+	2+	FB	5/4	3				-*****
ADDD	C3	4/3	3	D3	6/4	2	E3	6+/5+	2+	F3	7/5	3				-*****
*ADDE	@8B	3	3	@9B	5/4	3	@AB	5+	3+	@BB	6/5	4				
*ADDF	@CB	3	3	@DB	5/4	3	@EB	5+	3+	@FB	6/5	4				
*ADDW	!8B	5/4	4	!9B	7/5	3	!AB	7+/6+	3+	!BB	8/6	4				
*AIM				02	6	3	62	7+	3+	72	7	4				
ANDA	84	2	2	94	4/3	2	A4	4+	2	B4	5/4	3				--**0-
ANDB	C4	2	2	D4	4/3	2	E4	4+	2	F4	5/4	3				--**0-
ANDCC	1C	3	2													?????1
*ANDD	!84	5/4	4	!94	7/5	3	!A4	7+/6+	3+	!B4	8/6	4				
ASLA													48	2/1	1	--****
ASLB													58	2/1	1	--****
*ASLD													!48	3/2	2	
ASL				08	6/5	2	68	6+	2+	78	7/6	3				--****
ASRA													47	2/1	1	--****
ASRB													57	2/1	1	--****
*ASRD													!47	3/2	2	
ASR				07	6/6	2	67	6+	2+	77	7/6	3				--****
BITA	85	2	2	95	4/3	2	A5	4+	2+	B5	5/4	3				--**0-
BITB	C5	2	2	D5	4/3	2	E5	4+	2+	F5	5/4	3				--**0-
*BITD	!85	5/4	4	!95	7/5	3	!A5	7+/6+	3+	!B5	8/6	4				
*BITMD	@3C	4	3													
CLRA													4F	2/1	1	--0100
CLRB													5F	2/1	1	--0100
*CLRD													!4F	3/2	2	
*CLRE													@4F	3/2	2	
*CLRF													@5F	3/2	2	
*CLRW													!5F	3/2	2	
CLR				0F	6/5	2	6F	6+	2+	7F	7/6	3				--0100
CMPA	81	2	2	91	4/3	2	A1	4+	2+	B1	5/4	3				--****
CMPB	C1	2	2	D1	4/3	2	E1	4+	2+	F1	5/4	3				--****
CMPD	!83	5/4	4	!93	7/5	3	!A3	7+/6+	3+	!B3	8/6	4				--****
*CMPE	@81	3	3	@91	5/4	3	@A1	5+	3+	@B1	6/5	4				--****
*CMPF	@C1	3	3	@D1	5/4	3	@E1	5+	3+	@F1	6/5	4				--****
CMPS	@8C	5/4	4	@9C	7/5	3	@AC	7+/6+	3+	@BC	8/6	4				--****
CMPU	@83	5/4	4	@93	7/5	3	@A3	7+/6+	3+	@B3	8/6	4				--****
*CMPW	!81	5/4	4	!91	7/5	3	!A1	7+/6+	3+	!B1	8/6	4				--****
CMPX	8C	4/3	3	9C	6/4	2	AC	6+/5+	2+	BC	7/5	3				--****
CMPY	!8C	5/4	4	!9C	7/5	3	!AC	7+/6+	3+	!BC	8/6	4				--****

Mnem	Immed.			Direct			Indexed			Extended			Inherent			CCodes 53210
	OP	~/~	#	OP	~/~	+	OP	~/~	#	OP	~/~	#	OP	~/~	#	IHNZVC
COMA													43	2/1	1	--**01
COMB													53	2/1	1	--**01
*COMD													!43	3/2	2	
*COME													@43	3/2	2	
*COMF													@53	3/2	2	
*COMW													!53	3/2	2	
COM				03	6/5	2	63	6+	2+	73	7/6	3				--**01
CWAI	3C	22/20	2													E?????
DAA													19	2/1	1	--****
DECA													4A	2/1	1	--***-
DECB													5A	2/1	1	--***-
*DECD													!4A	3/2	2	
*DECE													@4A	3/2	2	
*DECF													@5A	3/2	2	
*DECW													!5A	3/2	2	
DEC				0A	6/5	2	6A	6+	2+	7A	7/6	3				--***-
*DIVD	@8D	25	3	@9D	27/26	3	@AD	27+	3+	@BD	28/27	4				
*DIVQ	@8E	34	4	@9E	36/35	3	@AE	36+	3+	@BE	37/36	4				
*EIM				05	6	3	65	7+	3+	75	7	4				
EORA	88	2	2	98	4/3	2	A8	4+	2+	B8	5/4	3				--**0-
EORB	C8	2	2	D8	4/3	2	E8	4+	2+	F8	5/4	3				--**0-
*EORD	!88	5/4	4	!98	7/5	3	!A8	7+/6+	3+	!B8	8/6	4				
EXG	1E	8/5	2													------
INCA													4C	2/1	1	--***-
INCB													5C	2/1	1	--***-
*INCD													!4C	3/2	2	
*INCE													@4C	3/2	2	
*INCF													@5C	3/2	2	
*INCW													!5C	3/2	2	
INC				0C	6/5	2	6C	6+	2+	7C	7/6	3				--***-
JMP				0E	3/2	2	6E	3+	2+	7E	4/3	3				------
JSR				9D	7/6	2	AD	7+/6+	2+	BD	8/7	3				------
LDA	86	2	2	96	4/3	2	A6	4+	2+	B6	5/4	3				--**0-
LDB	C6	2	2	D6	4/3	2	E6	4+	2+	F6	5/4	3				--**0-
LDD	CC	3	3	DC	5/4	2	EC	5+	2+	FC	6/5	3				--**0-
*LDE	@86	3	3	@96	5/4	3	@A6	5+	3+	@B6	6/5	4				
*LDF	@C6	3	3	@D6	5/4	3	@E6	5+	3+	@F6	6/5	4				
*LDQ	CD	5	5	!DC	8/7	3	!EC	8+	3+	!FC	9/8	4				
LDS	!CE	4	4	!DE	6/5	3	!EE	6+	3+	!FE	7/6	4				--**0-
LDU	CE	3	3	DE	5/4	2	EE	5+	2+	FE	6/5	3				--**0-
*LDW	!86	4	4	!96	6/5	3	!A6	6+	3+	!B6	7/6	4				
LDX	8E	3	3	9E	5/4	2	AE	5+	2+	BE	6/5	3				--**0-
LDY	!8E	4	4	!9E	6/5	3	!AE	6+	3+	!BE	7/6	4				--**0-
*LDMD	@3D	5	3													
LEAS							32	4+	2+							---i--
LEAU							33	4+	2+							---i--
LEAX							30	4+	2+							---i--
LEAY							31	4+	2+							---i--

Mnem	Immed.			Direct			Indexed			Extended			Inherent			CCodes 53210
	OP	~/~	#	OP	~/~	+	OP	~/~	#	OP	~/~	#	OP	~/~	#	IHNZVC
LSLA/LSLB/LSLD/LSL	Same as ASL															--0***
LSRA													44	2/1	1	--0***
LSRB													54	2/1	1	--0***
*LSRD													!44	3/2	2	
*LSRW													!54	3/2	2	
LSR				04	6/5	2	64	6+	2+	74	7/6	3				--0***
MUL													3D	11/10	1	---*-*
*MULD	@8F	28	4	@9F	30/29	3	@AF	30+	3+	@BF	31/30	4				
NEGA													40	2/1	1	-?****
NEGB													50	2/1	1	-?****
*NEGD													!40	3/2	2	
NEG				00	6/5	2	60	6+	2+	70	7/6	3				-?****
NOP													12	2/1	1	------
*OIM				01	6	3	61	7+	3+	71	7	4				
ORA	8A	2	2	9A	4/3	2	AA	4+	2	BA	5/4	3				--**0-
ORB	CA	2	2	DA	4/3	2	EA	4+	2	FA	5/4	3				--**0-
ORCC	1A	3/2	2													??????
*ORD	!8A	5/4	4	!9A	7/5	3	!AA	7+/6+	3+	!BA	8/6	4				
PSHS	34	5+/4+	2													------
PSHU	36	5+/4+	2													------
*PSHSW	!38	6	2													
*PSHUW	!3A	6	2													
PULS	35	5+/4+	2													??????
PULU	37	5+/4+	2													??????
*PULSW	!39	6	2													
*PULUW	!3B	6	2													
ROLA													49	2/1	1	--****
ROLB													59	2/1	1	--****
*ROLD													!49	3/2	2	
*ROLW													!59	3/2	2	
ROL				09	6/5	2	69	6+	2+	79	7/6	3				--****
RORA													46	2/1	1	--****
RORB													56	2/1	1	--****
*RORD													!46	3/2	2	
*RORW													!56	3/2	2	
ROR				06	6/5	2	66	6+	2+	76	7/6	3				--****
RTI													3B	6/17	1	-*****
														15/17		
RTS													39	5/4	1	------
SBCA	82	2	2	92	4/3	2	A2	4+	2+	B2	5/4	3				--****
SBCB	C2	2	2	D2	4/3	2	E2	4+	2+	F2	5/2	3				--****
*SBCD	!82	5/4	4	!92	7/5	3	!A2	7+/6+	3+	!B2	8/6	4				
SEX													1D	2/1	1	--**--
*SEXW													14	4	1	

Mnem	Immed.			Direct			Indexed			Extended			Inherent			CCodes 53210
	OP	~/~	#	OP	~/~	+	OP	~/~	#	OP	~/~	#	OP	~/~	#	IHNZVC
STA				97	4/3	2	A7	4+	2+	B7	5/4	3				--**0-
STB				D7	4/3	2	E7	4+	2+	F7	5/4	3				--**0-
STD				DD	5/4	2	ED	5+	2+	FD	6/5	3				--**0-
*STE				@97	5/4	3	@A7	5+	3+	@B7	6/5	4				
*STF				@D7	5/4	3	@E7	5+	3+	@F7	6/5	4				
*STQ				!DD	8/7	3	!ED	8+	3+	!FD	9/8	4				
*STS				!DF	6/5	3	!EF	6+	3+	!FF	7/6	4				
STU				DF	5/4	2	EF	5+	2+	FF	6/5	3				--**0-
*STW				!97	6/5	3	!A7	6+	3+	!B7	7/6	4				
STX				9F	5/4	2	AF	5+	2+	BF	6/5	3				--**0-
STY				!9F	6/5	3	!AF	6+	3+	!BF	7/6	4				--**0-
SUBA	80	2	2	90	4/3	2	A0	4+	2+	B0	5/4	3				--****
SUBB	C0	2	2	D0	4/3	2	E0	4+	2+	F0	5/4	3				--****
SUBD	83	4/3	3	93	6/4	2	A3	6+/5+	2+	B3	7/5	3				--****
*SUBE	@80	3	3	@90	5/4	3	@A0	5+	3+	@B0	6/5	4				
*SUBF	@C0	3	3	@D0	5/4	3	@E0	5+	3+	@F0	6/5	4				
*SUBW	!80	5/4	4	!90	7/5	3	!A0	7+/6+	3+	!B0	8/6	4				
SWI													3F	19/21	1	1-----
SWI2													!3F	20/22	2	E-----
SWI3													@3F	20/22	2	E-----
SYNC													13	2+/1+	1	------
TFR	1	1F	6/4	2												------
*TIM				0B	6	3	6B	7+	3+	7B	5	4				
TSTA													4D	2/1	1	--**0-
TSTB													5D	2/1	1	--**0-
*TSTD													!4D	3/2	2	
*TSTE													@4D	3/2	2	
*TSTF													@5D	3/2	2	
*TSTW													!5D	3/2	2	
TST				0D	6/4	2	6D	6+/5+	2+	7D	7/5	3				--**0-

Bit Transfer/Manipulation
AND,AND NOT, OR,OR NOT, ...: instr, post byte, memory location

Mnem	Direct		
	OP	~/~	#
*BAND	@30	7/6	4
*BIAND	@31	7/6	4
*BOR	@32	7/6	4
*BIOR	@33	7/6	4
*BEOR	@34	7/6	4
*BIEOR	@35	7/6	4
*LDBT	@36	7/6	4
*STBT	@37	8/7	4

Post-Byte

7	6	5	4	3	2	1	0

Bits 7 and 6: Register
00 - CC 10 - B
01 - A 11 - Unused
Bits 5, 4 and 3: Source Bit
Bits 2, 1 and 0: Destination bit

Source/Destination Bit in binary form:
0 - 000 2 - 010 4 - 100 6 - 110
1 - 001 3 - 011 5 - 101 7 - 111

Both the source and destination bit portions of the post-byte are looked at by the 6309 as the actual bit NUMBER to transfer/store. Use the binary equivalent of the numbers (0 thru 7) and position them into the bit area of the post byte. Ex: BAND A,1,3,240.

Branch Instructions

Mnem	OP	Mnem	OP	Description	Condition	Notes 1
BCC	24	LBCC	!24	Carry Clear	!C	M,U,4
BCS	25	LBCS	!25	Carry Set	C	M,U,5
BEQ	27	LBEQ	!27	Equal	Z	M,S,U
BGE	2C	LBGE	!2C	Greater Or Equal	N*V + !N*!V	S
BGT	2E	LBGT	!2E	Greater Than	N*V*!Z + !N*!V*!Z	S
BHI	22	LBHI	!22	Higher	!C*!Z	U
BHS	2F	LBHS	!2F	Higher Or Same	!C	U,4
BLE	2F	LBLE	!2F	Less Than Or Equal	Z + N*!V + !N*V	S
BLO	25	LBLO	!25	Lower	C	U,5
BLS	23	LBLS	!23	Lower Or Same	C + Z	U
BLT	2D	LBLT	!2D	Less Than	N*!V + !N*V	S
BMI	2B	LBMI	!2B	Minus (Negative)	N	M
BNE	26	LBNE	!26	Not Equal	!Z	M,S,U
BPL	2A	LBPL	!2A	Plus (Positive)	!N	M
BRA	20	LBRA	16	Always	1	O,2
BRN	21	LBRN	!21	Never	0	O
BSR	8D	LBSR	17	Subroutine	1	O,3
BVC	28	LBVC	!28	Overflow Clear	!V	M,S
BVS	29	LBVS	!29	Overflow Set	V	M,S

Short branches (column 1,2) have a signed byte destination [-128,127] range.
L prefixed long branches (column 3,4) have a signed word [-32768,32767] range.
Condition codes are untouched by branches.

Notes:
1 - Except notes 2,3, generic branch 6809/6309 cycles and byte lengths are in the table ->
2 - BRA and LBRA cycles in table ->
3 - BSR and LBSR cycles in table ->
4 - (L)BHS and (L)BCC are the same
5 - (L)BCS and (L)BLO are the same
S - Signed
U - Unsigned
M - siMple - tests single condition code.
O - other

Mnem	Immed. ~/~	#
B??	3	2
LB??	5/6	4
BRA	3	2
LBRA	5/4	3
BSR	7/6	2
LBSR	9/7	3

Register Descriptions, * Indicates new registers in 6309 CPU.

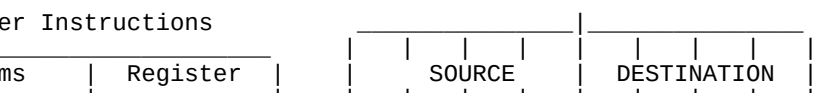
X - 16 bit index register	PC - 16 bit program counter register
Y - 16 bit index register	*V - 16 bit variable register
U - 16 bit user-stack pointer	*0 - 8/16 bit zero register
S - 16 bit system-stack pointer	V and 0 only inter-register instrcts
A - 8 bit accumulator	
B - 8 bit accumulator	Accumulator structure map:
*E - 8 bit accumulator	
*F - 8 bit accumulator	A | B | E | F
D - 16 bit concatenated reg.(A B)	
*W - 16 bit concatenated reg.(E F)	D | W
*Q - 32 bit concatenated reg.(D W)	
	Q
*MD - 8 bit mode/error register	
CC - 8 bit condition code register	31 24 15 8 0
DP - 8 bit direct page register	bit

Note: The 6309 is static, so the V register is saved across powerups! Others?

Transfer/Exchange and Inter-Register Post Byte

Inter-Register Instructions

Mnem	Forms	Register		
		OP	~/~	+
*ADCR	R0,R1	!31	4	3
*ADDR	R0,R1	!30	4	3
*ANDR	R0,R1	!34	4	3
*CMPR	R0,R1	!37	4	3
*EORR	R0,R1	!36	4	3
EXG	R0,R1	1E	8/5	2
*ORR	R0,R1	!35	4	3
*SBCR	R0,R1	!33	4	3
*SUBR	R0,R1	!32	4	3
TFR	R0,R1	1F	6/4	2
*TFM	R0+,R1+	@38	6+3n	3
*TFM	R0-,R1-	@39	6+3n	3
*TFM	R0+,R1	@3A	6+3n	3
*TFM	R0,R1+	@3B	6+3n	3



Register Field
(source or destination)

0000 - D (A:B)	1000 - A
0001 - X	1001 - B
0010 - Y	1010 - CCR
0011 - U	1011 - DPR
0100 - S	1100 - 0
0101 - PC	1101 - 0
0110 - W	1110 - E
0111 - V	1111 - F

TFM is Transfer Memory: repeats W times, decrementing W, changing Ri as asked. n in cycles is number of bytes moved.

Illegal to use CC, DP, W, V, 0, or PC as source or destination register.

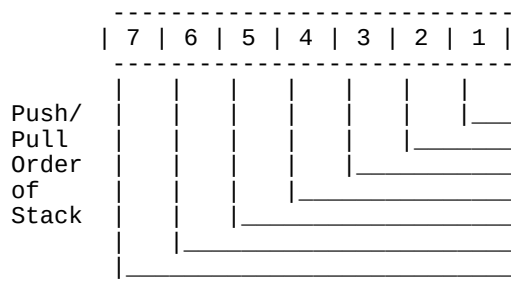
The results of all Inter-Register operations are passed into R1 with the exception of EXG which exchanges the values of registers and the TFR block transfers. The register field codes %1100 and %1101 are both zero registers. They can be used as source or destination.

Logical Memory Operations

AND,EOR,OR,TEST Immediate to memory: instr, post byte, operand

Mnem	Immed.			Direct			Indexed			Extended			Inherent		
	OP	~/~	#	OP	~/~	#	OP	~/~	#	OP	~/~	#	OP	~/~	#
*AIM				02	6	3	62	7+	3+	72	7	4			
*EIM				05	6	3	65	7+	3+	75	7	4			
*OIM				01	6	3	61	7+	3+	71	7	4			
*TIM				0B	6	3	6B	7+	3+	7B	5	4			

Push/Pull Post byte



All 2 byte registers are pushed low byte, then high byte. Stack grows down. The PSH(s,u) and PUL(s,u) instructions require one additional cycle for each byte pushed or pulled. A+B=D, E+F=W, W+D=Q, pushes low then high order. In 6309 mode interrupt stores 2 more bytes (E,F) on stack, and pops on RTI.

Push order --> PC, U/S, Y, X, DP, *F, *E/*W, B/D/*Q, A, CC <-- Pull order
On IRQ, all regs pushed. On 6309 mode, *W pushed after DP, before D.
FIRQ pushes only CC by default. On 6309 mode with FIRQ operating as IRQ, pushes W also. PS(U/S)W PUL(U/S)W saves/loads the W register.

Indexed and Extended Addressing Modes and Post byte Information

	Indexed				Extended		
Type	Forms	Asm form	+/+ ~/~	+ #	PostByte OP code	Asm form	+ ~	+ #
Constant offset from register R	No offset	,R	0	0	1rrY0100	[,R]	3	0
	5 bit offset	n,R	1	0	0rrnnnnn			
	8 bit offset	n,R	1	1	1rrY1000	[n,R]	4	1
	16 bit offset	n,R	4/3	2	1rrY1001	[n,R]	7	2
Accumulator offset from R (2's complement offset)	A - Register	A,R	1	0	1rrY0110	[A,R]	4	0
	B - Register	B,R	1	0	1rrY0101	[B,R]	4	0
	E - Register	E,R	1	0	*1rrY0111	[E,R]	1	0
	F - Register	F,R	1	0	*1rrY1010	[F,R]	1	0
	D - Register	D,R	4/2	0	1rrY1011	[D,R]	4	0
	W - Register	W,R	4/1	0	*1rrY1110	[W,R]	4	0
Auto increment and decrement of register R	Increment 1	,R+	2/1	0	1rrY0000			
	Increment 2	,R++	3/2	0	1rrY0001	[,R++]	6	0
	Decrement 1	,-R	2/1	0	1rrY0010			
	Decrement 2	,--R	3/2	0	1rrY0011	[,--R]	6	0
2's complement offset from PC	8 bit offset	n,PC	1	1	1xxY1100	[n,PC]	4	1
	16 bit offset	n,PC	5/3	2	1xxY1101	[n,PC]	8	2
Indirect	16 bit address				10011111	[n]	5	2
Rel to W 2's comp AutoIncr W AutoDecr W	No Offset	,W	0	0	*100ZZZZZ	[,W]	0	0
	16 bit offset	n,W	5/2	2	*101ZZZZZ	[n,W]	5	2
	Increment 2	,W++	3/1	0	*110ZZZZZ	[,W++]	3	0
	Decrement 2	,--W	3/1	0	*111ZZZZZ	[,--W]	3	0

* 6309 only. rr: 00 = X, 01 = Y, 10 = U 11 = S. xx: Doesn't care, leave 0.
Mode: Y = 0 index, Y = 1 extend; ZZZZZ = 01111 index, ZZZZZ = 10000 extend.
+ and +   indicates the additional number of cycles and bytes for the variation.
~   #

Condition Code Register (CC)

E	F	H	I	N	Z	V	C

Entire flag(7)
FIRQ mask(6)
Half carry(5)
IRQ mask(4)
Carry flag(0)
Overflow(1)
Zero(2)
Negative(3)

Mode and Error Register (MD, 6309 only)

?	?					?	?

Div by Zero(7)
Illegal Op(6)
Unused(5)
Unused(4)
Emulation Mode(0)
FIRQ Mode(1)
Unused(2)
Unused(3)

MD register: works like the CC register.
Bits 0,1 write only, bits 6,7 read only.
Bit 0: Emulation mode: if 0, 6809 emulation mode, if 1, 6309 native mode
Bit 1: FIRQ Mode   : if 0, FIRQ as normal 6809, if 1, FIRQ operate as IRQ
Bits 2-5 unused.
Bit 6: Set to 1 if illegal instruction occurred
Bit 7: Set to 1 if divide by 0 occurred
FIRQ saves only CC, unless in IRQ mode, then all registers in push order saved

6309/6809 Instructions (by opcode grid, transposed): (*prefix means 6309 only)
All unused opcodes are both undefined and illegal

L\H	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	NEG	pref	BRA	LEAX	NEG	NEG	NEG	NEG	SUBA	SUBA	SUBA	SUBA	SUBB	SUBB	SUBB	SUBB
x1	*OIM	pref	BRN	LEAY		*OIM	*OIM		CMPA	CMPA	CMPA	CMPA	CMPB	CMPB	CMPB	CMPB
x2	*AIM	NOP	BHI	LEAS		*AIM	*AIM		SBCA	SBCA	SBCA	SBCA	SBCB	SBCB	SBCB	SBCB
x3	COM	SYNC	BLS	LEAU	COM	COM	COM	COM	SUBD	SUBD	SUBD	SUBD	ADDD	ADDD	ADDD	ADDD
x4	LSR	*SEXW	BCC	PSHS	LSR	LSR	LSR	LSR	ANDA	ANDA	ANDA	ANDA	ANDB	ANDB	ANDB	ANDB
x5	*EIM		BCS	PULS		*EIM	*EIM		BITA	BITA	BITA	BITA	BITB	BITB	BITB	BITB
x6	ROR	LBRA	BNE	PSHU	ROR	ROR	ROR	ROR	LDA	LDA	LDA	LDA	LDB	LDB	LDB	LDB
x7	ASR	LBSR	BEQ	PULU	ASR	ASR	ASR	ASR		STA	STA	STA		STB	STB	STB
x8	ASL		BVC		ASL	ASL	ASL	ASL	EORA	EORA	EORA	EORA	EORB	EORB	EORB	EORB
x9	ROL	DAA	BVS	RTS	ROL	ROL	ROL	ROL	ADCA	ADCA	ADCA	ADCA	ADCB	ADCB	ADCB	ADCB
xA	DEC	ORCC	BPL	ABX	DEC	DEC	DEC	DEC	ORA	ORA	ORA	ORA	ORB	ORB	ORB	ORB
xB	*TIM		BMI	RTI		*TIM	*TIM		ADDA	ADDA	ADDA	ADDA	ADDB	ADDB	ADDB	ADDB
xC	INC	ANDC	BGE	CWAI	INC	INC	INC	INC	CMPX	CMPX	CMPX	CMPX	LDD	LDD	LDD	LDD
xD	TST	SEX	BLT	MUL	TST	TST	TST	TST	BSR	JSR	JSR	JSR	STD	STD	STD	STD
xE	JMP	EXG	BGT			JMP	JMP	LDX	LDX	LDX	LDX	LDU	LDU	LDU	LDU	LDU
xF	CLR	TFR	BLE	SWI	CLR	CLR	CLR	CLR		STX	STX	STX		STU	STU	STU
NOTES:					A	B	i	e		d	i	e	m	d	d	e

opcodes prefixed by 10

L\H	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0		LBRA	*ADDR	*NEGD					*SUBW	*SUBW	*SUBW	*SUBW				
x1		LBRN	*ADCR						*CMPW	*CMPW	*CMPW	*CMPW				
x2		LBHI	*SUBR						*SBCD	*SBCD	*SBCD	*SBCD				
x3		LBLS	*SBCR	*COMD	*COMW				CMPD	CMPD	CMPD	CMPD				
x4		LBHS	*ANDR	*LSRD	*LSRW				*ANDD	*ANDD	*ANDD	*ANDD				
x5		LBLO	*ORR						*BITD	*BITD	*BITD	*BITD				
x6		LBNE	*EORR	*RORD	*RORW				*LDW	*LDW	*LDW	*LDW				
x7		LBEQ	*CMPR	*ASRD						*STW	*STW	*STW				
x8		LBVC	*PSHSW	*ASLD					*EORD	*EORD	*EORD	*EORD				
x9		LBVS	*PULSW	*ROLD	*ROLW				*ADCD	*ADCD	*ADCD	*ADCD				
xA		LBPL	*PSHUW	*DECD	*DECW				*ORD	*ORD	*ORD	*ORD				
xB		LBMI	*PULUW						*ADDW	*ADDW	*ADDW	*ADDW		*LDQ		
xC		LBGE		*INCD	*INCW				CMPY	CMPY	CMPY	CMPY		*STQ		
xD		LBLT		*TSTD	*TSTW										LDS	LDS	LDS	LDS
xE		LBGT							LDY	LDY	LDY	LDY			STS	STS	STS	
xF		LBLE	SWI2	*CLRD	*CLRW					STY	STY	STY				
NOTES:				h	h				m	d	i	e	m	d	i	e

opcodes prefixed by 11

L\H	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0				*BAND					*SUBE	*SUBE	*SUBE	*SUBE	*SUBF	*SUBF	*SUBF	*SUBF
x1				*BIAND					*CMPE	*CMPE	*CMPE	*CMPE	*CMPF	*CMPF	*CMPF	*CMPF
x2				*BOR												
x3				*BIOR	*COME	*COMF				CMPU	CMPU	CMPU	CMPU			
x4				*BEOR												
x5				*BIEOR												
x6				*LDBT					*LDE	*LDE	*LDE	*LDE	*LDF	*LDF	*LDF	*LDF
x7				*STBT						*STE	*STE	*STE				
x8				*TFM												
x9				*TFM												
xA				*TFM	*DECE	*DECF										
xB				*TFM					*ADDE	*ADDE	*ADDE	*ADDE	*ADDF	*ADDF	*ADDF	*ADDF
xC				*BITMD	*INCE	*INCF				CMPS	CMPS	CMPS	CMPS			
xD				*LDMD	*TSTE	*TSTF			*DIVD	*DIVD	*DIVD	*DIVD				
xE									*DIVQ	*DIVQ	*DIVQ	*DIVQ				
xF				SWI2	*CLRE	*CLRF			*MULD	*MULD	*MULD	*MULD				
NOTES:									m	d	i	e	m	d	i	e

Notes:
A - operate on register A e - extended addressing m - immediate addressing
B - operate on register B h - inherent addressing
d - direct addressing i - indexed addressing

Mnemon.	Description	Notes	Mnemon.	Description	Notes
ABX	Add to Index Reg	X=X+B	LBcc nn	Long cond Branch	If cc LBRA
ADCa s	Add with Carry	a=a+s+C	LBRA nn	Long Br. Always	PC=nn
*ADCD s	Add with Carry	D=D+s+C	LBSR nn	Long Br. Sub	-[S]=PC,LBRA
*ADCR rr	add carry	r2=r2+r1+C	LDa s	Load acc.	a=s
ADDa s	Add	a=a+s	LDD s	Load D acc.	D=s
*ADDe s	Add	a=a+s	*LDe s	Load e acc.	e=s
ADDD s	Add to D acc.	D=D+s	*LDQ s	Load Q acc.	Q=s
*ADDR rr	Add registers	r2=r2+r1	*LDMD s	Load MD acc.	MD=s
ANDa s	Logical AND	a=a&s	LDS s	Load S pointer	S=s
ANDCC s	Logic AND w CCR	CC=CC&s	LDU s	Load U pointer	U=s
*ANDD s	Logical AND	D=D&s	LDi s	Load index reg	i=s (Y ~s=7)
*ANDR rr	Logical AND regs	r2=r2&r1	LEAp s	Load Eff Address	p=EAs(X=0-3)
ASL d	Arith Shift Left	d=d*2	LSL d	Logical Shift L	d={C,d,0}<-
ASLa	Arith Shift Left	a=a*2	LSLa	Logical Shift L	a={C,a,0}<-
*ASLD	Arith Shift Left	D=D*2	*LSLD	Logical Shift L	D={C,D,0}<-
ASR d	Arith Shift Rght	d=d/2	LSR d	Logical Shift R	d=->{d,0}
ASRa	Arith Shift Rght	a=a/2	LSRa	Logical Shift R	d=->{d,0}
*ASRD	Arith Shift Rght	D=D/2	*LSRD	Logical Shift R	D=->{W,0}
BCC m	Branch Carry Clr	If C=0	*LSRW	Logical Shift R	W=->{W,0}
BCS m	Branch Carry Set	If C=1	MUL	Multiply	D=A*B
BEQ m	Branch Equal	If Z=1	*MULD s	Multiply	Q=D*s
BGE m	Branch >=	If NxV=0	NEG d	Negate	d=-d
BGT m	Branch >	If Zv{NxV}=0	NEGa	Negate acc	a=-a
BHI m	Branch Higher	If CvZ=0	*NEGD	Negate acc	D=-D
BHS m	Branch Higher,=	If C=0	NOP	No Operation	
BITa s	Bit Test acc	a&s	ORa s	Logical incl OR	a=avs
*BITD s	Bit Test acc	D&s	ORCC n	Inclusive OR CC	CC=CCvn
*BITMD s	Bit Test acc	MD&s	*ORD s	Logical incl OR	D=Dvs
BLE m	Branch <=	If Zv{NxV}=1	*ORR rr	Logical incl OR	r1=r1vr2
BLO m	Branch Lower	If C=1	PSHS r	Psh reg(s)(!= S)	-[S]={r,...}
BLS m	Branch Lower,=	If CvZ=1	PSHU r	Psh reg(s)(!= U)	-[U]={r,...}
BLT m	Branch <	If NxV=1	*PSHSW	Psh reg W	-[S]=W
BMI m	Branch Minus	If N=1	*PSHUW	Psh reg W	-[U]=W
BNE m	Branch Not Equal	If Z=0	PULS r	Pul reg(s)(!= S)	{r,...}=[S]+
BPL m	Branch Plus	If N=0	PULU r	Pul reg(s)(!= U)	{r,...}=[U]+
BRA m	Branch Always	PC=m	*PULSW	Pul reg W	W=[S]+
BRN m	Branch Never	NOP	*PULUW	Pul reg W	W=[U]+
BSR m	Branch to Sub	-[S]=PC,BRA	ROL d	Rotate Left	d={C,d}<-
BVC m	Branch Over. Clr	If V=0	ROLa	Rotate Left acc.	a={C,a}<-
BVS m	Branch Over. Set	If V=1	*ROLD	Rotate Left acc.	D={C,D}<-
CLR d	Clear	d=0	*ROLW	Rotate Left acc.	W={C,W}<-
CLRa	Clear acc.	a=0	ROR d	Rotate Right	d=->{C,d}
*CLRD	Clear acc.	D=0	RORa	Rotate Right acc	a=->{C,a}
*CLRe	Clear acc.	e=0	*RORD	Rotate Right acc	D=->{C,W}
CMPa s	Compare	a-s	*RORW	Rotate Right acc	W=->{C,W}
CMPD s	Compare D acc.	D-s	RTI	Return from Int	{regs}=[S]+
*CMPe s	Compare e acc.	e-s	RTS	Return from Sub	PC=[S]+
*CMPR rr	Compare regs	r1-r2	SBCa s	Sub with Carry	a=a-s-C
CMPS s	Compare S ptr	S-s	*SBCD s	Sub with Carry	D=D-s-C
CMPU s	Compare U ptr	U-s	*SBCR rr	Sub with Carry	r1=r1-r2-C
CMPi s	Compare	i-s (Y ~s=8)	SEX	Sign Extend	D=B extended
COM d	Complement	d=~d	*SEXW	Sign Extend	Q=W extended
COMa	Complement acc.	a=~a	STa d	Store accumultor	d=a
*COMD	Complement acc.	D=~D	STD d	Store Double acc	D=a
*COMe	Complement acc.	e=~e	*STe d	Store accumultor	d=e
CWAI n	AND CC, Wait int	CC=CC&n,E=1,	*STQ d	Store accumultor	d=Q
DAA	Dec Adjust Acc.	A=BCD format	STS d	Store Stack ptr	S=a
DEC d	Decrement	d=d-1	STU d	Store User ptr	U=a
DECa	Decrement acc.	a=a-1	STi d	Store index reg	i=a (Y ~s=7)
*DECD	Decrement acc.	D=D-1	SUBa s	Subtract	a=a-s
*DECe	Decrement acc.	e=e-1	SUBD s	Subtract D acc.	D=D-s
*DIVD s	Divide	D=D/s	*SUBe s	Subtract D acc.	e=e-s
*DIVQ s	Divide	Q=Q/s	*SUBR rr	Subtract regs	r1=r1-r2
EORa s	Logical Excl OR	a=axs	SWI	Software Int 1	-[S]={regs}

*EORD s	Logical Excl OR	D=Dxs	SWI2	Software Int 2	SWI
*EORR rr	Logical Excl OR	r1= r1xr2	SWI3	Software Int 3	SWI
EXG rr	Exchg(same size)	r1<->r2	SYNC	Sync. to int	(min ~s=2)
INC d	Increment	d=d+1	*TFM tf	Block transfer	- special-
INCa	Increment acc.	a=a+1	TFR r,r	Transfer r1->r2	r2=r1
*INCD	Increment acc.	D=D+1	TST s	Test	s
*INCe	Increment acc.	e=e+1	TSTa	Test accumulator	a
JMP s	Jump	PC=EAs	TSTD	Test accumulator	D
JSR s	Jump to Sub	-[S]=PC,JMP	TSTe	Test accumulator	e

a	Acc A or B	***** Legend - todo - do more *******	
e	Acc E, F, or W (6309)	* prefix	6309 only instruction
d s EA	Dest/Src/effective addr.	m	Rel addr (-128 to +127)
i p r	X orY/X,Y,S,U/any reg	n nn	8/16-bit (0 to 255/65535)
rr	two registers r1,r2	tf	transfer registers and +-

Interrupt Vectors

	FFF0 to FFF1	Note 1	FFF8 to FFF9	IRQ vector	
Reserved	FFF2 to FFF3	SWI3 vector	FFFA to FFFB	SWI vector	
Addresses	FFF4 to FFF5	SWI2 vector	FFFC to FFFD	NMI vector	
	FFF6 to FFF7	FIRQ vector	FFFE to FFFF	Reset vector	

Note 1: Reserved in 6809. For 6309 mode, holds vector for divide by 0 error or illegal instruction error. Error can be read in 6309 register MD.

The Hitachi HD63B09EP (6309) microprocessor is a clone of the Motorola MC68B09E (6809) chip, with additional registers and instructions. Bit 0 of the 6309 only register MD determines which mode is on: 6809 emulation or 6309 native. 6309 often has faster instruction timings. When cycle counts are given for 6809/6309 for a 6309 only instruction, these are for emulation/native timings.

The Motorola 6809 was released circa 1979, and came in many flavors: 68A09, 68A09E,68B09,68B09E. The 68A09(E) ran at 1 MHz and 1.5 MHz, the 68B09(E) at 2 MHz. The 6809 had an internal clock generator needing only an external crystal, and the 6809E needed an external clock generator.

The 6309 has a B (2 MHz) and a C version rated at either 3.0 or 3.5 MHz. Some hackers have pushed the 63C09 variant can to 5 MHZ. The 6309 comes in internal and external clock versions (HD63B/C09 and HD63B/C09E respectively).

Some useful code ideas, based on [1]: (see [1] for more info)
1. Check if code on a 6309 or 6809:

```
      LDB   #255  , CLRD    ; executes as a $10 (ignored) $4F (CLRA) on a 6809
      TSTB        , BEQ   Is6309
```

2. Check if 6309 system is in native mode or to check 6309 FIRQ mode, use RTI with appropriate items on stack.

Document History
- April 2007 - Version 1.1 - extensive additions, minor corrections.
- July 2006 - Version 1.0 - initial release.

Sources

[1] HD63B09EP Technical Reference Guide, Chet Simpson, Alan DeKok
[2] Programming the 6809, Rodney Zaks, William Labiak,1982 Sybex
[3] Notes by Jonathan Bowen.
[4] Notes by Neil Franklin, 2004.11.01
[5] Notes by Paul D. Burgin
[6] Notes by Sockmaster(John Kowalski)
[7] The MC6809 Cookbook, Carl Warren, 1981.
[8] en.wikipedia.org/wiki/6809
[9] www.howell1964.freeserve.co.uk/

END OF FILE